Programming Life - Final Report

Group: TAGC / PL5

Members:

Kasper Grabarz	4083245
Thomas Oomens	4422597
Jeffrey Helgers	4318749
Youp Mickers	4246713
Matthijs Rijlaarsdam	4308417

Deliverable Date:

23-6-2016

Table of contents

1 Introduction

Ever since DNA sequencing methods were first developed in the 1970's, DNA analysis has had a great impact on many different fields such as medicine, evolutionary biology, molecular biology, metagenomics and forensics. While DNA sequencing allows researchers to identify the precise order of nucleotides within DNA, it does not help them with the process of deriving useful information from these molecules. To aid them in this process a variety of tools exist already. However, current tools for *visualizing* genetic data do not meet all the requirements of "The Broad Institute". Their main requirements can be summed up as follows:

- To interactively navigate through the data with the aid of a phylogenetic tree;
- To be able to semantically zoom through the data;
- Mutation and annotation visualization and analysis;
- Connecting to external data sources such as PubMed;
- Being able to query the graph.

An application that is able to do all this does not exist.

The software product that was developed to try and address these requirements is called *Genan.* This tool is able to read genome assemblies stored in files that use the GFA (Graphical Format Assembly) format. It then uses a graph visualisation algorithm to draw this graph on the screen into a predefined viewport, giving the user a more intuitive and clearer view of the data. The user can navigate through the graph using either a minimap or a main view, using controls on the screen or a mouse. A user can view one genome or compare multiple genomes using a phylogenetic tree view. *Genan* can help researchers to find portions of interest and relations between other genomes based on current genetics theory.

2 Product overview

2.1 Module overview

Genan follows a modular design and has been structured as three loosely coupled interacting modules. These modules are the back end, a server module and the front end. The front end is implemented as a browser user interface and is the module that is responsible for what the user will see and interact with. The server module is responsible for instantiating a server that the browser interacts with through a RESTful API. It processes requests from the browser and passes them on to the back end module and then serves whatever response the back end generates back to the browser. The server module and the backend are written in Java and the front end is written in JavaScript, HTML and CSS.

2.2 Front end

The front end is located in the resource folder of the server module. It gets served by the server module after using the browser to go the address and the right port of the machine *Genan* is running on. By default *Genan* uses port 9998.

2.3 Server module

When the server module receives a request it delegates these to the back end module by calling the appropriate functions exposed through the back end interface. It then converts the result returned by the backend to a JSON object and sends this back to the browser. The server module implements a RESTful API by making use of the Jersey framework. Jersey is an implementation developed by Oracle of the JAX-RS (Java API for RESTful Web Services) API. The module runs on top of a Grizzly server which makes use of Java's non-blocking I/O APIs that offer features for intensive I/O operations.

2.4 Back end

This module contains all the algorithms and data structures used to compute the various functions that *Genan* offers to the user. One of its functions is the parsing of user provided data such as the genome graph, phylogenetic tree and other genome related data. Other functions it has are generating the graph and tree that the user sees in the front end based on user input, detecting mutations and possible evolutionary convergence, and searching the graph for genomic features.

3 Reflection on the product and process

3.1 Tools

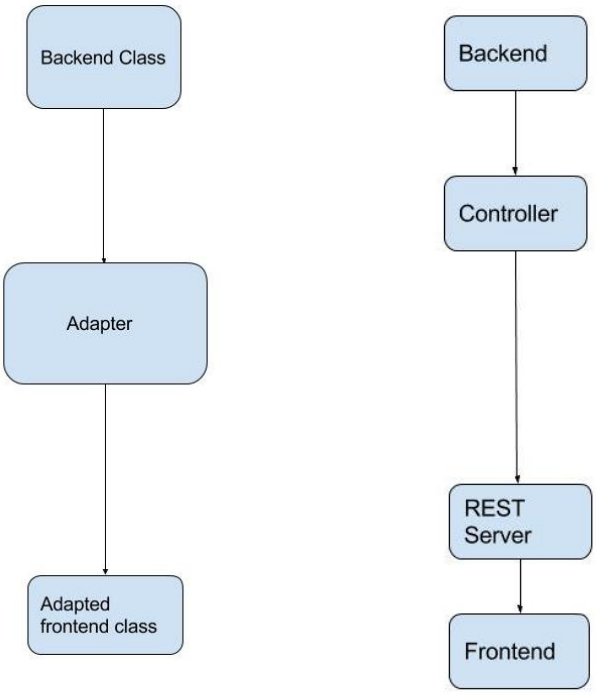
The product was developed as a Maven project. Maven helped with maintaining a structured build process and integration of various analysis and build process related plugins. The plugins used were Cobertura, Travis, Findbugs, Checkstyle and PMD of which the latter three provide static analysis functionality. For source code revisioning and management development was done using GitHub integrated with Travis. Travis is a plugin which enables automated testing each time code is pushed to the GitHub repository. Cobertura was used to compute test coverage and helped with making sure that the code was properly tested. Findbugs was used to detect bugs in the code that the java compiler sometimes misses or that become apparent only at runtime. The PMD and Checkstyle plugins helped with detecting code quality flaws such as overly complex code, code duplication, improper naming, and missing comments. All the changes to the code was pushed to the master repository through pull requests made by the person applying the changes. An important rule used was that always a person other than the one writing the new code had to approve the pull request. This practice made sure there was always more than one person responsible for a change. Travis helped with this by indicating if the new code pushed to the GitHub repository still passed all the tests. Using tooling in this way assured all code written was of sufficient quality.

3.2 Development process

To make sure the software development process followed a structured pattern the Scrum software development framework was used. This meant that each friday a new sprint backlog and a sprint retrospective reflecting on the previous sprint was written. During these meetings the team would decide which member would be doing what task for the coming week. The meetings were also used for evaluating the progress made so far. By allowing room for changes in the plan for the coming week, flexibility was provided that allowed the developers to not feel too constrained by their weekly sprint. This decision was consciously made, as the innovative and complex nature of the product meant not every addition took as much time to implement as was planned beforehand.

3.3 Software design

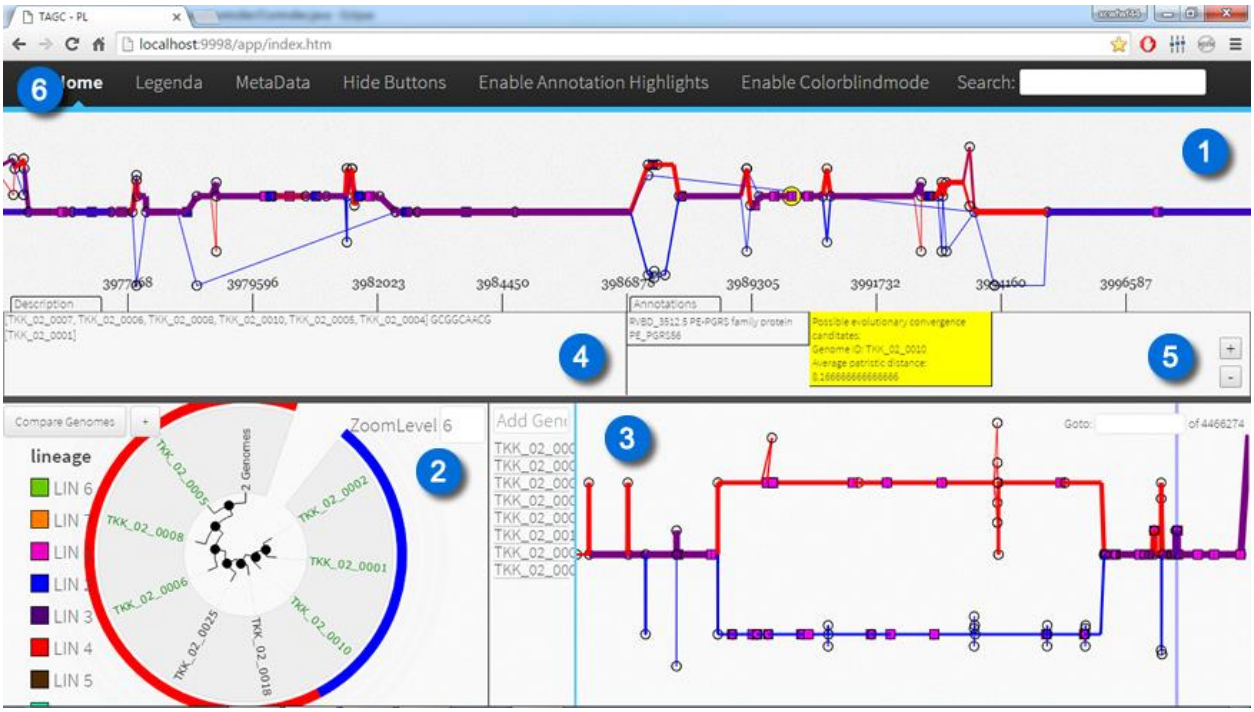
The product was made using several design patterns that made sure the code was clear, efficient and maintainable. The adapter pattern was used to make the interface used by the frontend and the interface provided by the backend independent of each other. Furthermore, due to the way the server is implemented, the adapter follows a singleton pattern so that the REST api of the server can call the functions that are part of the adapter.



The application as a whole also followed the MVC pattern. There is a controller class that connects the the model classes that make up the backend and the view that is served by the server. *Genan's* code follows the single responsibility principle by keeping the responsibilities of each class limited. The functions in each class were kept short and readable. All code is commented, making sure an external reader can easily understand the functionality covered by each function. The checkstyle plugin along with its configuration file helped spot any parts of the code that violated code quality criteria.

4 Product description

The GUI elements described in this section are visible in figure 1 below. *Genan* consists of three main views. Firstly there is a main view which is a detailed view of the genome graph being analyzed 1. Secondly there is a phylogenetic tree view which is used to select the genomes the user wants to analyze 2. Lastly there is a minimap which can be used to navigate through the graph and give a high level indication of position in the graph 3. Just below the main view there is a description view and an annotations view. These views display information for a node that the user has hovered over with their cursor. The description view displays the genomes sharing that node and the DNA sequence represented by it 4. The annotations view displays any genomic features contained within the DNA sequence and in case the node is part of a possible convergent mutation it will show a yellow box 5. This box will show the candidate genomes for possible evolutionary convergence and the average patristic distance of these genomes to the other genomes containing the node. At the top of the application window there is a top bar that contains a search box and multiple buttons with various functionalities to be described below 6.



4.1 Genome graph and phylogenetic tree viewing

Genan can parse and process genome graph files in the GFA format. It uses a phylogenetic tree defined in a file with the NWK format to draw the phylogenetic tree widget. The user can select genomes in the tree and press the compare genome button which then causes our application to process the genome graph and generate a minimap and main view of the selected genomes extracted from the graph. When completely zoomed in this newly created graph contains the same structures and relations that the genomes exhibit in the original

genome graph. The different zoom levels in this graph correspond to different tree levels in the phylogenetic tree. At each zoom level only the genome graph nodes are shown which are shared at that level in the tree. Specifically, for each subtree rooted at a tree node on a tree level only those genome graph nodes are shown that are shared between the genomes in the leafs of those trees. Using this phylogenetic filtering also allows the user to select only parts of the genome that entire families share, by selecting a middle node in the phylogenetic tree.

4.2 Genome feature searching and viewing

The user can supply an additional file with genomic features in the GFF file format. These features will then be parsed and can be searched for using their display name. The search function allows user to type in only partial matches to these names and our application will find any names that contains the substring. After clicking on a result *Genan* will automatically change its main view to zoom in to the node containing this feature.

4.3 Mutation identification

4.3.1 INDEL and SNP mutations

Mutations are automatically discovered and displayed in the gui. The mutations are made visible by showing their acronyms such as SNP for single nucleotide polymorphism or INDEL for insertion or deletion mutations. They are also given a distinct colour so the user can easily spot these mutations in the GUI. For the meaning of the various symbols one can refer to the legenda which can be accessed by clicking a link at the top of the application.

4.3.2 Convergent mutations

Apart from INDEL and SNP mutations, *Genan* can identify mutations that are potentially the result of convergent evolution. It surrounds such mutation with a yellow circle. One such mutation can be seen in the screenshot below. The algorithm compares the genomes that contain the mutation and looks for genomes which have an unusually large patristic distance compared to the other genomes. After clicking on such a mutation a yellow box will be displayed in the annotations view with the genomes that are candidate for convergent evolution and their average patristic distance to the other genomes with that mutation.

4.4 Metadata

Apart from the GFF, GFA and NWK files our product can also deal with genome metadata supplied by the user. The metadata must be in a simple table with header format and supplied as a flat file. This metadata can then be viewed by the user at runtime by selecting one of the metadata types in the top bar. After clicking on a type the colors in the heat map surrounding the phylogenetic tree will correspond to the categories comprising the metadata type. The legend for these colors can be seen to the left of the phylogenetic tree.

4.5 Navigation

4.5.1 Mouse or touchpad.

Using the mouse or touch pad you can either click and drag in the main view or minimap to move left or right in the genome graph. To zoom the user can use the scroll function of either the mouse or touchpad.

4.5.2 Goto function

With the goto function a user can type in nucleotide coordinates for which the axis can be seen at the bottom of the main view. After clicking enter *Genan* will zoom into that location.

4.5.3 Zoom buttons

The zoom buttons can be enabled or disabled using a button in the top bar. These zoom buttons allow users without a mouse or scroll functionality on their laptop to zoom in or out.

4.6 Visual aid

4.6.1 Annotation Highlights

There is a button in the top bar for toggling annotation highlighting. This feature encircles nodes in the graph that contain a genomic feature and thus helps the user spot these nodes. After click on the node the feature description will be visible in the annotations view below the main view.

4.6.2 Colorblind mode

For people with colorblindness there is a button in the top bar that can be pressed to toggle between two different color layouts. In colorblind mode the colors red and green will be replaced by other more suitable colours.

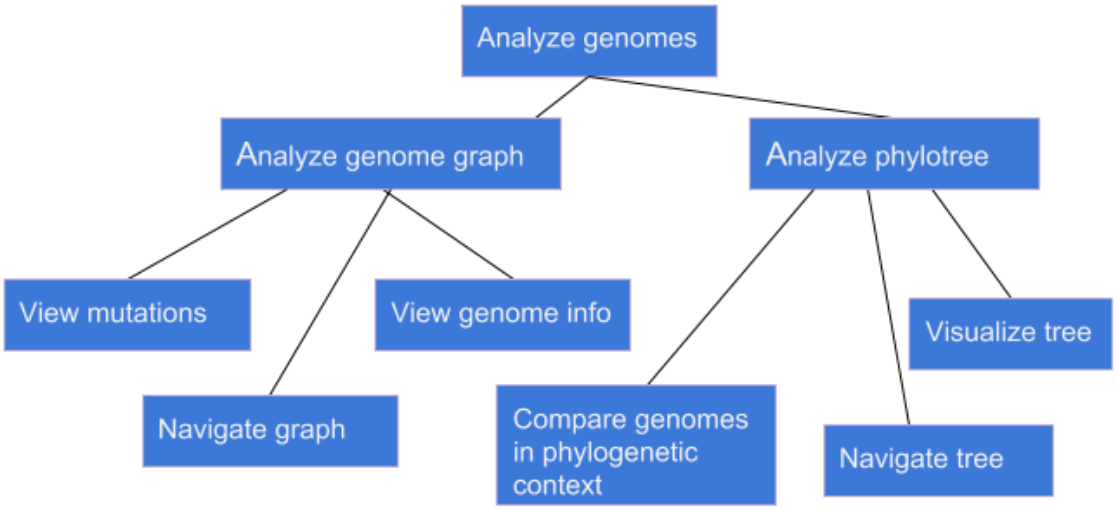
5 HCI analysis

In this section the human computer interaction (HCI) aspect of *Genan* is covered. It starts out with a user analysis in which a hypothetical user (persona) is described followed with some typical tasks this user might carry out. The latter part is presented through a task analysis diagram. It continues with a contextual inquiry where the context in which the product will be used is described and also in what way it is needed. Lastly a user evaluation of our product that has been performed is presented

5.1 User analysis

Persona: A genome researcher at the Broad Institute.

Task analysis:



5.2 Contextual Inquiry

Genome researchers need a tool to visualize and analyze large sets of genomic data. The goal of this is to find genes and other genomic features that are relevant from a biological point of view. The goal in this particular case is to analyze the genomes of bacteria that are responsible for human diseases with the purpose of better understanding their resistance to antibiotics, with the ultimate goal of developing a cure. To help with this analysis they need a way to visualize and navigate through large genome graphs and put them in a phylogenetic context. An important part of this analysis is being able to find and identify genes and mutations.

5.3 Usability evaluation

Setup

For the usability evaluation a thinking out loud session was organized. Persons were asked to give their initial thoughts of the program, after they were given a very brief explanation of resistance, genomes and what the program could be used for (see appendix A). They were asked to perform several tasks and were not given any guidance, unless they got completely stuck. Afterwards, a small interview was held in which the user answered several questions, while being able to access the application. In total, 5 users were used for the test, all in their early twenties. One of the persons was from a parallel group doing the same project, while the others had no prior knowledge of the project and no experience in genetics beyond high school biology. It was noted that the person from the parallel group was significantly faster in completing the tasks than the others, indicating that our program relies on specific knowledge and might therefore have a control flow that is more logical than indicated by general population testing.

The tasks they were asked to perform:
Initial thoughts about the layout

Enable color blind mode
Use the minimap to go to base/nucleotide 20000
Find out which genomes are resistant to Rifampin
    Change the number of visible levels of the phylogenetic tree.
Compare a genome which is resistant to Ofloxacin with one which is not
    Which of the genomes are selected?
    Find a SNP and mention which of the genomes has which nucleotide
    Find an INDEL
    Select genome TKK_03_0160 as well
Find a gene which is responsible for "cystathionine beta-lyase"
    What is the sequence of this gene?

5.4 Results

Menu bar

Almost all of the users (4/5) tried using the legend as a menu, which is a clear indication that the visual cues are incorrect for this feature. Some testers also noted that they wanted the legend to be visible at all times, although we expect this to change during prolonged use of the program. Two of the users indicated that "enable colorblindmode" and "enable annotation highlights" were not logically placed in the menu bar and should be given a less prominent place on the screen. When asked about this, two more users agreed that the positioning was confusing. Three of the users repeatedly clicked "hide/slow buttons" while looking around the screen, with some of them even asking what changed. When asked if the button was necessary, all of the users indicated that it was not.

Genome selection and phylogenetic tree

After clicking compare genomes, three of the users continued clicking on other buttons, indicating that the feature is too slow and a loading bar is necessary. When asked to select another genome, most of the users tried searching for it in the top-right search bar, instead of using the "add genome"-field. Seeing that genomes do not necessarily clash with annotations, we suggest that the two search bars get merged and adding genomes can also be done through the top-right search bar. One user mentioned that the nodes in the phylogenetic tree should be a bit larger, which was backed up by two other users during the interview session. When asked about the phylogenetic tree and the metadata visualization, all users were favorable about the representation. One user suggested that instead of directly deleting a genome if it was clicked in the "selected genome" view, it should show metadata, while deleting should be done via a red cross next to the genomes.

Minimap and general view

Resizing the different windows caused difficulties with most testers, as the area in which the border could be grabbed was very small. Two users did not notice that the views were draggable, while one other user seemed to try resizing the windows, only to give up after a few seconds. One of the users suggested that the minimap should cause a selection if it was clicked and dragged, which was met with favorable responses after other users were asked about this

potential feature. Three of the users said zooming in the main view was problematic, due to the fact that the mouse position was not taken as the exact center of zooming (giving priority to fixing this known bug).

Other suggestions
The +-button for increasing the size of the phylogenetic tree caused confusion if clicked and was generally not received positively. Another way of implementing this feature should be considered. Alphabetizing the metadata could be an option as one user said, which seems viable, given that selecting a certain resistance took considerable time, even after the user found the metadata menu. Another possibility would be make the metadata types searchable in the search bar. Finally, one user mentioned that the “compare genomes”-button would be more logically placed if it was directly above or below the selected genomes, a placement that would indeed give a better context.

6 Evaluation and failure analysis

Genan’s development provided various learning opportunities. One of the major bottlenecks in the development of the application was the fact that not the entire team had experience making a webapp. This made the decision to build a Web based application, based on the assumption that this would provide the opportunity to build a better frontend, less successful. Since not everyone was able to work on the frontend, there was often functionality implemented in the backend and not visible in the frontend. This slowed down productivity.

6.1 Backend module

The backend module generally functions very well. The parsing and processing of a genome graph of up to 328 genomes happens within 5 seconds on an Intel Core i7 with 4GB. However, *Genan* has difficulties with comparing large amounts of genomes at the same time, as the phylogenetic filtering is inefficient for such tasks. It does however provide the user with a way to compare genome families.

6.2 Server module

The server module implemented as a Jersey instance running in a Grizzly server is very simple and efficient. It allows for easy maintenance and expansion of the REST api. However, the marshalling of objects to JSON remains problematic and often requires a lot of boilerplate code to get it to work. With Jersey it is possible to freely chose a marshaller that implements the JAXB api and we have chosen Moxy. An alternative would be Jackson. The problem with Jackson is that often requires more annotations in the objects that need to be marshalled which would make the code less clean by entangling it with the JSON marshalling code.

6.3 Browser GUI

The browser gui is aesthetically pleasing and intuitive to work with. The functionality provided by our application is readily visible and using the text labels and conventional symbols a user should find its way through our app quite easily. The fact that the application can be interacted

with through a browser can be convenient for situations when users would like to access their data and the applications functionality remotely. Any libraries used by the GUI are provided by the server. This makes sure that an internet outage can not cause our application to fail as long as the user is located on the same local network as the server.

7 Outlook

There are four main points of improvement for the future:

Firstly, most members of the group often finished their weekly code contributions a day or less before we had to hand in our demo. This caused a lot of stress due to the little time that was left to integrate all the changes together. It also left little time for proper code review.

Secondly, the fundamentals data structures we used to represent the genome graph and for the node filtering algorithm often underwent changes throughout the course of the project. Because of this other classes and algorithms that used these data structures would stop working properly and would have to be rewritten. More emphasis should be placed next time on proper planning and designing of the data structures to prevent them from having to be changed too much.

Thirdly, in order to prevent the problems mentioned in the second point, classes should be designed in such way that changes to one will not affect the other. This could still be improved next time and would also reduce the problem mentioned in the second point.

Lastly, a few members sometimes felt they could not continue their code until another one finished theirs. This was usually because the code they worked on somehow interacted. This can be solved by both parties involved initially agreeing on an interface which defines how the code they work on will interact. Even though this strategy was discussed during the project it was not always properly followed by all the group members.

Appendix A: Handout during usability tests

Introduction

Antibiotics used to be the silver bullet in fighting bacterial infections. Unfortunately, resistance against anti-biotics is now wide-spread and on the rise amongst bacteria, threatening the effectiveness of anti-biotics and posing a serious threat to public health. The clue to resistance can be found in the blue-prints of the bacteria themselves, the DNA. As such, this program is meant as a tool for simplifying the comparing of two genomes. This in turn can result in a better understanding of resistance itself.

Glossary

Nucleotide – The smallest piece of information in a genome, can be considered the “bit” of DNA (possible values A, C, T, G).

Gene – Small part of the genome, often responsible for a single task (e.g. creation of certain enzyme) or property (e.g. hair colour) of the organism. Consists of multiple nucleotides.

Annotation – Label attached to genes to show their function/effect.

Genome – the genetic material (DNA-sequence) of a single organism. Can be regarded as the blueprint of each organism. Consists of multiple genes.

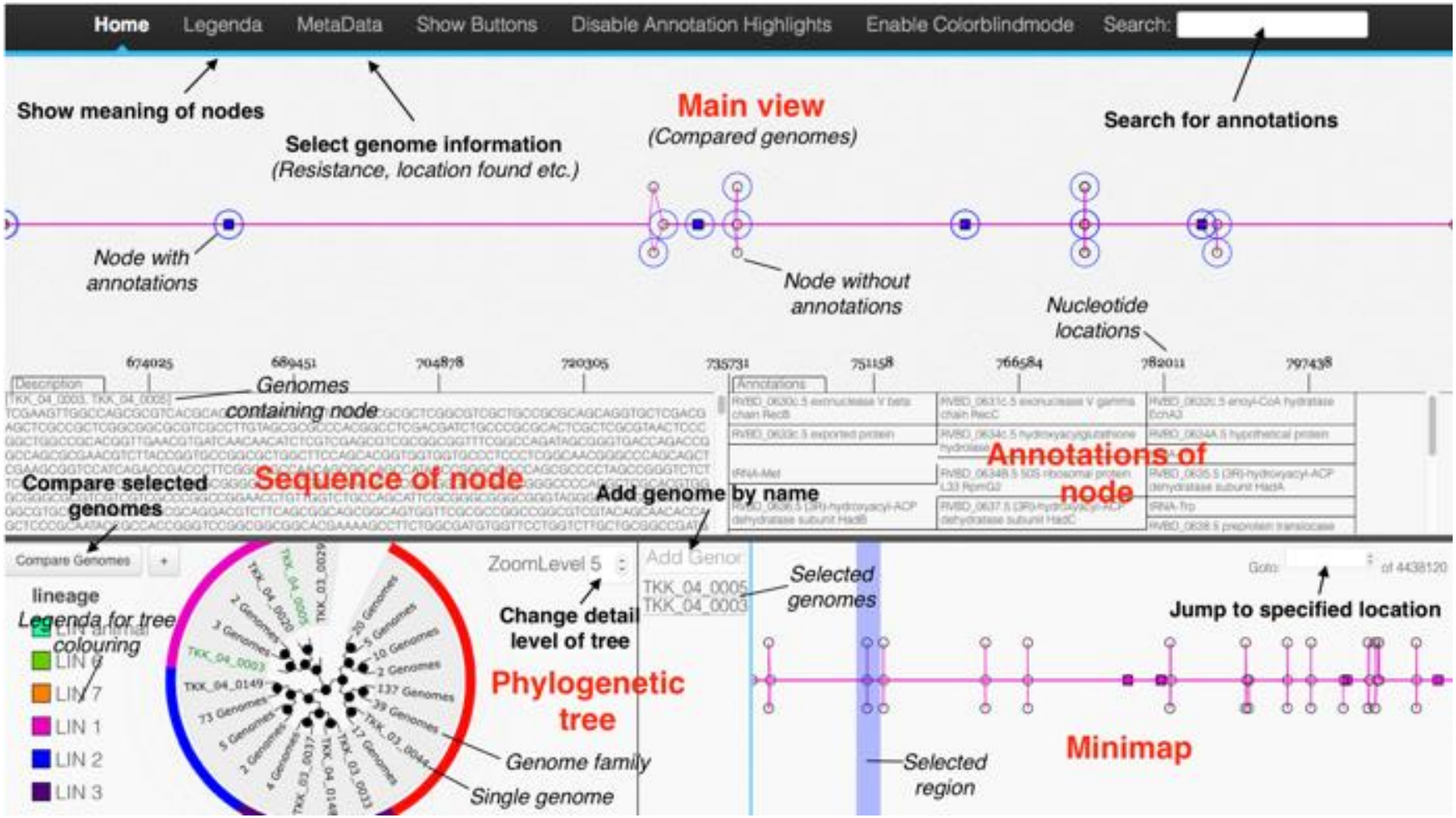
Phylogenetic tree – Tree relating different organisms based on similarities (similar organisms are close together). Can be compared to a family tree.

SNP – Single Nucleotide Polymorphism, a single nucleotide differing between two genomes.

INDEL – A single piece of DNA inserted in the genome, compared to another genome.

Strand – Small part of the genome, comparable to a gene, but arbitrary and without defined function. Contains a **sequence** of nucleotides.

Metadata – Included data of a genome/organism (e.g. whether it is resistant or where it was found).


Home
Legenda
MetaData
Show Buttons
Disable Annotation Highlights
Enable Colorblindmode
Search:
Show meaning of nodes
Select genome information
(Resistance, location found etc.)
Main view
(Compared genomes)
Search for annotations
Node with annotations
Node without annotations
Nucleotide locations
674025
689451
704878
720305
735731
751158
766584
782011
797438
Genomes containing node
Sequence of node
Annotations of node
Compare selected genomes
Add genome by name
Compare Genomes
ZoomLevel 5
Add Genor
TKK_04_0005
TKK_04_0003
Selected genomes
Goto:
of 4438120
Jump to specified location
lineage
Legenda for tree colouring
LIN 6
LIN 7
LIN 1
LIN 2
LIN 3
Change detail level of tree
Phylogenetic tree
Genome family
Single genome
Minimap
Selected region