




# Genome Complexity Browser: visualization and quantification of genome variability

Alexander Manolov<sup>1</sup><sup>\*</sup>, Dmitry Konanov<sup>1</sup>, Dmitry Fedorov<sup>1</sup>, Ivan Osmolovsky<sup>1</sup>, Elena Ilina<sup>1</sup>

**1** Federal Research and Clinical Centre of Physical and Chemical Medicine, Federal Medical and Biological Agency of Russia, Moscow, Russian Federation

 These authors contributed equally to this work. \* manolov@rcpcm.org

## Abstract

Comparative genomics studies may be used to acquire new knowledge about genome architecture - the rules to combine a set of genes in a genome of a living organism. Hundreds of thousands of prokaryote genomes were sequenced and assembled. Still, there is a lack of computational tools able to compare a large number of genomes simultaneously. We developed Genome Complexity Browser, a tool that allows visualization of gene contexts in a graph form and quantification of the variability of different parts of a genome. Graph visualization allows inspecting changes in gene content and neighborhood in hundreds of genomes simultaneously. This may be useful to find conservative and variable parts of operons or to estimate the overall variability of particular genome locus. We introduce a measure called complexity to quantify genome variability. Intraspecies and interspecies comparisons reveal that regions with high complexity values tend to be located in a conservative context in different strains and species.

## Author summary

A comparison of genomes of different bacteria and archaea species revealed that many of them frequently exchange genes. Occasionally such horizontal gene transfer events result in the acquisition of pathogenic properties or antibiotic resistance by the recipient organism. Previously it was observed that the probabilities of gene insertions are not equally distributed along a chromosome. There are loci called hotspots, in which changes occur with much higher frequencies than in other parts of the chromosome. We developed a computational method and a software tool called Genome Complexity Browser, that allows identification of genome variability hotspots and visualization of occurred changes. We performed a comparison of localization of hotspots and revealed that some of them have conservative localization even when comparing different species, while others are transient. Our tool allows users to visually inspect patterns of gene changes in a graph form, which makes such visualizations compact and informative.

## Introduction

Although highly variable, prokaryotic genomes do not represent simply a set of genes, they possess regularities, which are collectively termed "genome architecture" [1–3]. Rules of genome architecture can shed light on still unknown molecular mechanisms

1

2

3

4

governing prokaryotic cell life and may be essential in the engineering of synthetic organisms.

Non-random localization of different genes may be important due to several factors. Genes located near the replication origin have higher copy numbers in fast-dividing cells – a so-called replication-associated gene dosage effect [4, 5]. Folding of the chromosome makes genes located in different parts of the chromosome close to each other in 3D space, which can be beneficial for the gene coding for a regulator and its targets [6, 7]. Effects of global regulators (such as H-NS) on gene expression were observed to depend on the location of the target gene [8] and transcriptional propensity also varies depending on the chromosome position [9]. Cooperative effects between RNA polymerases [10] or supercoiling propagation effects may play a role in the transcriptional regulation of neighboring genes.

It was observed that horizontal gene transfer (HGT) events are preferentially localized in hotspots – chromosomal loci in which changes are observed much more frequently than in other regions [11–13]. This might indicate that, although disruptions in genome architecture may result in decreased fitness of an organism, there are some places in the chromosome where changes can be introduced without inducing negative effects. To our knowledge, there is no currently available computational tool to perform quantitative estimation of genome variability along the chromosome. Such a tool is needed to deepen our knowledge of factors determining genome variability and stability.

Here we present Genome Complexity Browser (GCB), the tool that allows estimation of local genome variability and visualization of gene rearrangements. Local genome variability is estimated using a graph representation of gene order (neighborhood) with a here introduced measure called complexity. Complexity profiles may be used to identify hotspots of horizontal gene transfer or other gene rearrangement events. Graph-based visualization available in GCB allows analysis of patterns of genome changes events and detection of persistent or variable gene combinations (i.e. variable and conservative parts of operons).

## Materials and methods

### Graph construction

Input to this step is the set of genomes with inferred orthology groups. The algorithm for graph construction is the following: each orthologous group is represented as a node, and two nodes are connected by a directed edge if the corresponding genes are located sequentially in at least one genome in a set. The weight of the edge is calculated as the number of genomes in which corresponding genes are adjacent (see Fig 1 A,B). Graph objects and their methods are implemented in gene-graph-lib library for Python 3, more information can be found in the library documentation at [https://github.com/DNKonanov/gene\\_graph\\_lib](https://github.com/DNKonanov/gene_graph_lib).

Because GCB uses directed graph representation of gene order, all genomes in a set should first be coaligned to achieve the same orientation throughout the set. The algorithm for this step is listed in S1 Listing).

Orthology groups inference is a difficult task and inferred groups often include paralogues genes [14]. Two methods are suggested in GCB to resolve such situations. The first method is a simple deletion of all groups which have more than one representative. An advantage of this approach is simplicity and clear output, while some genes are missed in the graph (S1 Fig A). The second approach is not to skip paralog genes, but to “orthologize” them. Each group with a unique context is designated with a unique suffix and becomes a node in the graph (S1 Fig B). GCB uses the first approach by default, while the second approach is available as an option both in the

**Fig 1. Principal scheme of a graph representation of gene order in a set of genomes and genome variability estimation approach.** To construct a graph, each orthology group is represented as a node. Nodes are connected by a directed edge if the corresponding genes are arranged sequentially in at least one genome in the set. A. Genomes 1,2,3 represent three different hypothetical genomes. Arrows represent genes, and genes within one orthologous group have the same color and letter designation. B. Graph representation of three genomes shown in A). The weight of the edge (arrow width) is calculated as the number of genomes in which corresponding genes are located sequentially. C. Deviating paths for node X are defined as paths in the graph which bypass the node X and are connected with the section of the reference node chain limited by the window parameter. D. Two examples of counting deviant paths are shown. X is the considered node, deviating paths are shown with blue lines. Complexity value is defined by the number of deviating paths.

command line and browser-based versions.

## Genome complexity definition

We introduce a measure called complexity for quantification of a local genome variability. Complexity values are calculated against one genome from the set that can be selected arbitrarily. This genome is extracted from the graph as a simple chain of nodes that is called the reference chain (Fig 1C). To calculate complexity in the node X, nodes from the reference chain in the range  $\pm window$  around the node X are selected, and complexity is defined as the number of distinct paths in the graph that do not contain the node X but start and finish in the nodes from the selected range (deviating paths), see Fig 1C.

Complexity computing is an iterative algorithm that generates a set of possible deviating paths from each node in the reference genome (Algorithm1) and when a new unique deviating path is found, the algorithm adds  $1/window$  to complexity value of all nodes in the reference between nodes coinciding with the start and the end of the deviating path (Algorithm2).

The algorithm has the following user-defined parameters: *window* - the size of the area around node X to which deviating paths should be connected (default 20 nodes), *iterations* - number of random walk processes from each node (default 500)

## Simulations of genomes with predefined variability profiles.

We supposed that calculated complexity value in some region of a chromosome correlates with the frequency of fixed rearrangements in that region. To verify this assumption and validate the algorithm, sets of model genomes were generated. These simulations included random gene insertions, deletions, HGT events, and inversions. HGT and random insertion probabilities were chosen to be equal to deletion events probability to maintain genome length. The probability of inversion was chosen as 1/100 than others, in agreement with literature data that inversion events are less common than other types of rearrangements, such as deletions and duplication [15]. The localization of these changes through chromosome was chosen according to the predefined variability profile. Next, these model genomes were processed by the complexity computing algorithm and results were compared with input distributions (more details are available in S1 Text). The algorithm for simulations is available in S2 Listing.

---

**Algorithm 1: FIND PATHS**

---

**Data:** graph, start\_node, ref\_chain, iterations

**Result:** Paths - set of all paths, which start in start\_node and end the in ref\_chain

$Paths \leftarrow$  empty set

$i \leftarrow 1$

**while**  $i \leq$  iterations **do**

$next\_node \leftarrow$  select random node connected with start\_node

$path \leftarrow [ \text{start\_node} ]$

**while**  $next\_node$  **not in** ref\_chain **do**

**if**  $next\_node$  **not in** path **do**

            add  $next\_node$  to path

$next\_node \leftarrow$  select random node connected with the last node in the path

**if** all nodes connected with  $next\_node$  are in path **do**

$path \leftarrow [ ]$

**break**

**if** length(path) > 1 **and** path **not in** Paths **do**

        add path to Paths

$i \leftarrow i + 1$

**return** Paths

---

---

**Algorithm 2: COMPUTE COMPLEXITY**

---

**Data:** graph, reference organism, window, number of iterations

**Result:** complexity values for each node in the reference

$ref\_chain \leftarrow$  reference nodes chain

set initial complexity values for all nodes as 0

**for each** node **in**  $ref\_chain$  **do**

$Paths \leftarrow$  FIND PATHS(graph, node, ref\_chain, number of iterations)

**for each** path **in** Paths **do**

$start \leftarrow$  first node in the path

$end \leftarrow$  last node in the path

$distance \leftarrow | \text{position}(start) - \text{position}(end) |$

**if**  $distance \leq$  window **do**

**for each** node between start and end **do**

$complexity[node] \leftarrow complexity[node] + 1/\text{window}$

**return** complexity values

---

## Subgraph generation

To visualize a gene context in the region of interest, a subgraph representing this region is constructed. First, a subset of reference chain nodes representing the region of interest is added to the graph. Next, the algorithm iterates through other genomes from the set and adds deviating paths limited to the selected region. If the length of the path is greater than the *depth* parameter, then the path is cropped and only start and end fragments (tails) of a fixed length (*tails* parameter, *tails* < *depth*) are added to the subgraph. If the weight of some edge is less than the user-defined *minimal\_edge\_weight* parameter, this edge is not added to the subgraph. The subgraph generation algorithm is listed in S3 Listing.

## Web server data acquisition and preparation

To construct a dataset for the web server, we downloaded genomes for 143 prokaryotic species that had more than 50 genomes available in the RefSeq database. For each species, if the number of complete genomes available was higher than 50, then only complete genomes were used. If the number of available genomes was higher than 100, then exactly 100 genomes were randomly selected for further analysis. The only exception was *Escherichia coli* extended genome set, which contained 327 complete genomes available as of November 2017.

All downloaded genomes were reannotated with Prokka ver 1.11 [16] to achieve uniformity. Genes were assigned to orthologous groups with OrthoFinder ver. 2.2.6 [17]. Python scripts that are a part of the GCB application were used to parse OrthoFinder output, calculate genome complexity values and generate subgraphs around genome regions of interest.

## Additional methods

Phylogenetic trees were inferred with Parsnp v1.2 [18]. Retention indexes were calculated using RI function from R phangorn library [19]. To estimate similarity to the reference genome in the analysis presented in Figure 6, all genomes were aligned with nucmer [20] and similarity score was calculated as follows: all aligned reference genome ranges were reduced with IRanges R package [21] and their total length was divided by reference genome length, all query genomes were sorted by this value and strains with the top 100 highest values were chosen. Nucmer was used to detect synteny blocks between genomes from the same species, and Mauve [22] was used to detect synteny blocks between genomes from different species. Prophages were detected with Phaster [23]. To obtain Figure 3A we used GCB with following parameters *tails* = 1, *minimal\_edge* = 5. To produce Figure 3B we used GCB with following parameters *tails* = 1, *minimal\_edge* = 5. To produce Figure 3C we used GCB with the following parameters: *window* = 20, *tails* = 0, *minimal\_edge* = 5. Code to make Figures 4-6 is available at <https://github.com/paraslonic/GCBPaperCode>.

## Results

### Software description and availability

The GCB tool is available as a standalone application and as a web server. GCB web server uses precalculated data for 143 species and is available at <http://gcb.repcm.org>. The standalone browser-based application and a set of command-line scripts are available at <https://github.com/DNKonanov/GCB>.

GCB browser-based GUI consists of three main parts: 1) the top panel that allows selection of a genome and a region to work with, 2) the complexity plot showing complexity profile for selected genome and contig (in case if the assembly contains more than one contig), 3) the subgraph visualization form (Figure 2).

**Fig 2. Screenshot of GCB browser-based interface.** GCB GUI is available at gcb.repcm.org and when GCB is run as a local server. It consists of top panel to select organism, complexity profile panel, subgraph visualization panel, nodes and edges information panel.

Several settings are implemented to customize subgraph visualization to make it convenient for the analysis. Subgraph can also be exported in JSON format and visualized with specialized software (i.e. Cytoscape) to prepare publication-ready images.

For analysis of a custom group of organisms (i.e. species absent on a web server) we provide a command-line Snakemake script to infer homology groups, Python3 scripts to obtain a text file with complexity values and a database file, that can be imported to the local GCB server. Local GCB server can be run on a standard PC.

To estimate the genomic variability profile, the number of genomes should not be too small, a few dozens or hundreds are typical values. The upper limit depends on the computational resources available to infer homology groups, which is the most computationally difficult step. Further details and instructions are provided in the user manual available at [https://github.com/DNKKononov/Genome-Complexity-Browser/blob/master/GCB\\_manual.pdf](https://github.com/DNKKononov/Genome-Complexity-Browser/blob/master/GCB_manual.pdf).

### Subgraph visualization

Graph representation of gene order provides a convenient way to inspect visually the context of genes of interest and to identify conservative and variable gene combinations. GCB can construct and visualize subgraph - part of the graph containing the region of interest. Next, we describe examples of subgraphs generated with GCB and visualized with Cytoscape [24].

#### Subgraph visualization reveals conservative and variable parts of operons.

Fig 3A shows the subgraph representing the gene context of the capsule gene cluster (chromosomal coordinates 3111444-3128026 in NCBI sequence NC\_011993.1) in 327 complete genomes of *Escherichia coli*. From this visualization, it can be seen that the operon has two conservative parts and one variable part. Variable part consists of genes from the serotype-specific synthesis region while neighboring conservative parts correspond to regions involved in polysaccharide export [25]. Capsule is considered as important virulence factor [26] for *E. coli* and many other species. Its variation is essential in avoidance of immune response and phage infection [27,28].

While the existence of variable and conservative parts of this operon was previously known, new information about the architecture of other operons may be obtained with such type of analysis that can be done with GCB.

#### Subgraph visualization reveals a genome variability in a particular locus

Fig 3 B, C shows the visualization of subgraphs of regions containing two operons: hemin uptake (hmu) and propanediol utilization (pdu). The occurrence of *E. coli* harboring these operons in the intestinal microbiome was previously shown to be associated with Crohn's disease [29–31]. These operons have different phylogenetic

**Fig 3. Subgraph visualization allows identification of conservative and variable parts of operons and reveals genome variability in a particular locus.** Subgraphs of the genomic regions in which A) capsule gene cluster, B) hemin uptake and C) propanediol utilization operon are located. A) graph visualization reveals the conservative and variable parts of the capsule gene operon. Visualizations in B) and C) indicate that both operons are located in a conservative context. Genomes that do not contain hemin uptake operon do not contain other genes in the same context. In the case of propanediol utilization operon, in several genomes alternative and highly variable gene sets are present.

distribution: hmu operon is preferentially present in B2 phylogroup (S2 Fig A, retention index = 1), pdu operon can be found in phylogenetically distinct strains of *E. coli*, and its presence is in low agreement with the phylogenetic tree (S2 Fig B, retention index = 0.26). Hmu operon is located on the 3691615-3700567 positions and pdu operon on the 2083448-2101340 positions of NCBI Reference Sequence NC\_011993.1).

The graph visualization reveals that hmu operon is located in a conservative context, which means that the neighboring orthologous groups are the same in all strains in which it is present (Fig 3 B). The edge that bypasses the operon indicates that in some genomes the genes to the left and right of the operon are adjacent. Graph visualization also indicates that one of the genes (hemin transport system permease, HmuU) or its close homologs are present in two alternative contexts.

Pdu operon is harbored by only a fraction of all considered strains (27 out of 327), but also located in a conservative context (Fig 3 C). Some variations in pdu operon are visible and reflect different operon variants [31]. Unlike the case of hemin uptake locus, here alternative gene sets are present. These alternative sets include genes of iron transport (FepC, FcuA, HmuU), DNA mobilization (retroviral integrase core domain, transposase DDE Tnp ISL3). What is especially interesting is the high variability of that alternative sets, with many overlapping changes being visible in the subgraph visualization.

In the next section, we will describe a quantitative measure of this observable difference in complexity of subgraphs.

## Complexity is a measure of genome variability

In a set of genomes with identical gene content and localization, each node in the resulting graph will have two edges. Any gene rearrangements (deletion, translocation, insertion) result in the addition of new edges. We hypothesized that the number of distinct paths in a subgraph representing a genomic region will monotonically depend on the frequency of fixed gene rearrangements in that region.

We implemented the algorithm (Algorithm 1,2 in Methods) to count the number of distinct random walks in a subgraph representing a given region of the reference genome, the value which we further call complexity of the region. By selecting subregions with the sliding window and calculating complexity value for them we get the complexity profile of the reference genome. The size of the sliding window can be set by the user, the default value of 20 was used for the results described below.

## Method verification

We verified our approach by performing simulations of genome evolution through gene transfer events, by comparison with the previously developed method and by analysis of known variability hotspot regions (integrations).

Complexity profile for *E. coli* calculated with GCB is in good agreement with hotspots evaluated in [32], see Fig 4A. Complexity values within hotspots identified in [32] are significantly greater than outside of the hotspots (p-value  $<10^{-16}$ , Mann-Whitney test, see Fig 4B).

**Fig 4. Complexity value can be used as a genome variability measure.** A) Comparison of complexity profile for *E. coli* K12 with hotspots identified in [32] (blue rectangles underneath complexity profiles). B) Comparison of complexity values for genes inside and outside hotspots from [32]. C) Comparison of initial variability profiles with complexity profiles calculated from genomes after 3000 evolution simulation steps. D) Complexity profile for *Vibrio cholerae* N16961 chromosome II with noticeable high levels at the integron region.

We performed a number of simulations in which we suggested that the probability of genomic rearrangement events (HGT, deletion, translocation) is non uniformly distributed along the chromosome. This may reflect the unequal probability of changes or their fixation. The algorithm to make simulations is listed in S2 Listing. We used three patterns to generate profiles of such probabilities: sinusoidal, rectangular, and sawtooth and performed 10 independent simulations for each pattern. The number of rearrangement iterations was 3000 for each model. The results of our method were in good correspondence with the predefined distribution (R-square  $>0.74$ , Spearman correlation  $>0.69$ , FDR corrected p-value  $<10^{-300}$ ), see Fig 3C. Results for each simulation are available in S1 Text.

Finally, integrons have expectedly high complexity values computed with here proposed method. Fig 3D shows as example integron region of *V. cholerae*, which is a known source of this species diversity and was dubbed superintegron because of its high length of about 120 kbp and more than a hundred of integrated cassettes [33].

### High complexity values associated with prophages and genomic islands

Fig 5 shows the complexity profile along the chromosome of adherent-invasive *E. coli* LF82 [34]. As expected, the regions with a higher density of essential genes have low complexity values. On the contrary, pathogenicity islands and prophage regions have relatively high complexity values. At the same time, there are chromosomal loci with relatively high complexity values that have no recognizable signs of mobile elements. A subgraph of the region with the highest complexity values (located at 2,115,791:2,164,382 chromosomal positions) do not contain recognizable genes with mobility associated functions.

**Fig 5. Regions with high complexity values are mostly associated with mobile elements.** Complexity profile for the *E. coli* LF82 chromosome is shown, 327 other *E. coli* genomes were used to calculate complexity values. Orange color bars denote prophages, red color bars denote pathogenicity islands. For essential genes, low complexity values are observed. Some most variable regions lack features of mobile elements.

### Complexity profiles have conservative features on the inter- and intraspecies levels

The proposed method of genome complexity analysis can be used to compare variability profiles of different species and intraspecies structures (i.e. phylogroups). Interspecies comparisons for the 146 species used in the current study reveals that when genomes



are similar enough (synteny blocks covers most part of the genomes), then complexity profiles have many common features, i.e. regions with high complexity values in different genomes are located in conservative context (S3 Fig).

Fig 6A shows comparison of complexity profiles of the four *Bacillus* species, Fig 6B for their phylogenetic relation. It can be seen that regions with high complexity values are associated with prophages (denoted with an orange bar below the complexity profile) and have conservative location, similar to *E. coli*. Some regions that lack integrated viruses also have high complexity values and conservatively located in genomes of different species (i.e. the one located at 2.5 Mbp in *B. subtilis*), while others are only highly variable at one species only (i.e. the one located at 2.8 Mbp in *B. velezensis*).

**Fig 6. Regions with high complexity values are mainly located in a conservative context, when both intra and interspecies comparisons are performed.** A) Complexity profiles and synteny blocks of the four *Bacillus* species; B) phylogenetic tree of the four *Bacillus* species; C) complexity profiles and synteny blocks of the five *E. coli* phylogroups; D) phylogenetic tree of the *E. coli* genomes selected for the analysis, for each phylogroup one reference strain and 100 closest genomes was selected.

Fig 6C shows a comparison of complexity profiles for different *E. coli* phylogroups [35]. For each of the five large phylogroups (A, B1, B2, D, E) we selected one reference strain and 100 most similar strains from 5466 RefSeq genomes (both finished and draft assemblies), see Fig 6D for the phylogenetic tree of selected genomes. Complexity profiles for each reference genome were inferred using genomes from corresponding clade only. This comparison reveals that many of the regions with high variability rate are located in the same context in the genomes of the strains belonging to the different phylogroups. The majority of them contain prophages, but some do not include phage-associated genes. Transient hotspots (with high complexity in some clades and low complexity values in others) can also be observed.

## 1 Discussion

Synteny visualization tools (i.e. Mauve [22], BRIG [36], genePlotR) are often used for genome comparative studies. Such tools allow visual inspection of large and small genome rearrangements. The number of genomes which can be effectively compared with such approach ranges from a few to several dozens. Meanwhile, hundreds and even thousands of genomes are available for some species now. This large amount of genomes may be used to gain new information about genome variability, genome architecture, and structure of operons. To analyze efficiently large sets of genomes we propose a graph-based approach, in which genes are represented with nodes that are connected depending on their co-localization (neighborhood).

Graphs were previously applied to analyze genome changes in a form of breakpoint graphs [37], which is useful to reconstruct possible ancestral states but, in our opinion, are not convenient for visualization properties. They also were used to represent known genome variants to increase mapping quality [38]. To our knowledge, gene neighborhood graph visualization is available only in FindMyFriends R package beside GCB.

Graph representation of a set of genomes and selecting a subgraph representing a region of interest facilitates answering the following questions. Is a gene (operon) located in the same context in all genomes? If not, then what alternatives are present? Which parts of a gene set (operon) are conservative and which are variable? Which genomes contain some particular combination of genes?

Hotspots of genome variability were described for a number of bacterial species. In [32] the authors analyzed HGT hot spots for 80 bacterial species. They concluded that many hotspots lack mobile genetic elements and proposed that homologous recombination is mainly responsible for the variability of those loci. The factors that determine the location of hot spots, their emergence, and elimination, are still an open question.

We implemented a method for quantification of local genome variability based on the number of unique paths in a subgraph. To our knowledge, it is the first tool that allows quantification of genome variability based on a user-defined set of genomes. GCB provides a way to study dynamics of variability hot spots, changes in their intensity and location on different levels ranging from intraspecies structures like phylogroups or ecotypes to interspecies and intergenus comparisons.

We compared variability profiles between different species and, in the case of *E. coli*, between different phylogroups. We observed that, as a rule, when genomes are close enough for the large syntenic blocks to be detected (with blast or nucmer tool), then complexity profiles look similar: the regions with high complexity values are surrounded with low complexity regions forming the same conservative context in different groups of organisms. The analysis of complexity profiles of *E. coli* revealed that many hotspots are located in the prophage or pathogenicity islands integration sites, and site-specific mechanisms could govern their conservative location. Some hotspots lack such factors and reasons for their conservative location are still to be elucidated.

The here proposed approach is not universal, for example, it is not suited for the detection of large genomic rearrangements (larger than window parameter, usually several dozens genes) or changes in noncoding parts of the genome. Our methodology has also some limitations coming from its dependence on orthology inference accuracy. Here we used orthofinder tool [17], which uses MCL graph clustering algorithm based on gene length normalized blast scores. We find this tool to be optimal in terms of efficiency and accuracy. On the other hand, it doesn't take into account phylogenetic information and syntenic relationships between different genomes, and erroneous homology inference sometimes occurs. Paralogous genes may be attributed to one group. In this case, the graph representation of the context becomes problematic. We implement two possible ways of dealing with paralogous genes in GCB: the default approach is to ignore them, the other is to perform an artificial orthologization process (each paralogous gene with unique left and right context is denoted with a suffix and added to the graph). From our experience, the optimal strategy is to work in the default mode for explorative analysis and verify all conclusions in the orthologization mode. The graph layout process is also hard to automate. We use two layout algorithms (Dagre and Graphviz), but manual manipulations are often needed to make a clear layout, and Cytoscape (or other graph manipulation software) is desirable to make publication-ready images.

Despite the above-mentioned drawbacks, we find the here proposed method of complexity analysis informative as it successfully identifies known rearrangement hot spots (prophages, integrons et al.), and we hope that GCB with its capacity of visualization and complexity assessment will find its application in the area of comparative genomics studies.

## Conclusion

We have developed a novel tool, called Genome Complexity Browser (GCB), to analyze sets of genomes in order to quantify genome variability and visualize gene context variants. We use graph representation of genes neighbourhood to make visualizations and estimate local genome variability. GCB browser-based interface enables

simultaneous analysis of genome variability profile and pattern of changes occurred in a particular locus.

We precalculated data for 143 prokaryotic organisms and web server gcb.rcpcm.org can be used to browse them. Command line tool and stand-alone server application make possible for user to analyze any particular set of genomes.

We observed that there are genome regions with high variability which have conservative localization in intraspecies and interspecies comparisons. Some of them are free of genes with identifiable mobility functions.

## Supporting information

**S1 Fig. Comparison of two methods to deal with the paralogs.** A) The graph obtained with the default approach to ignore groups with several representatives in the particular genome; B) graph obtained with paralogs “orthologization” approach.

**S2 Fig. Phylogenetic tree of 327 *E. coli* strains with pdu and hmu operon presence information.** Red bars denote genomes in which complete gene set from the operon is present, green bars denote genomes in which more than the half of operon genes are present. A) Hmu operon is in good correspondence with the phylogenetic tree of *E. coli*; B) pdu operon presence is poorly correlated with the phylogenetic tree of *E. coli*.

**S3 Fig. Comparison of complexity profiles of 146 prokaryotic species.** Complexity profiles are shown on the same scale for all organisms. Synteny blocks are shown in green color. The phylogenetic tree was built based on 16S rRNA sequence.

**S1 Text Simulations of genomes with predefined variability profiles.** Details concerning method and results of genome simulations with predefined variability profile with further complexity analysis.

**S1 Listing. Algorithm to coalign genomes in the set.**

**S2 Listing. Algorithm to simulate genomes with predefined variability distribution.**

**S3 Listing. Algorithm to generate subgraph representing genome region of interest.**

## References

1. Rocha EP. The organization of the bacterial genome. Annual review of genetics. 2008;42:211–233.
2. Touchon M, Rocha EP. Coevolution of the organization and structure of prokaryotic genomes. Cold Spring Harbor perspectives in biology. 2016;8(1):a018168.
3. Hendrickson HL, Barbeau D, Ceschin R, Lawrence JG. Chromosome architecture constrains horizontal gene transfer in bacteria. PLoS genetics. 2018;14(5):e1007421.

4. Couturier E, Rocha EP. Replication-associated gene dosage effects shape the genomes of fast-growing bacteria but only for transcription and translation genes. *Molecular microbiology*. 2006;59(5):1506–1518.
5. Slager J, Veening JW. Hard-wired control of bacterial processes by chromosomal gene location. *Trends in microbiology*. 2016;24(10):788–800.
6. Dorman CJ. Genome architecture and global gene regulation in bacteria: making progress towards a unified model? *Nature Reviews Microbiology*. 2013;11(5):349.
7. Fritsche M, Li S, Heermann DW, Wiggins PA. A model for Escherichia coli chromosome packaging supports transcription factor-induced DNA domain formation. *Nucleic acids research*. 2011;40(3):972–980.
8. Brambilla E, Sclavi B. Gene regulation by H-NS as a function of growth conditions depends on chromosomal position in Escherichia coli. *G3: Genes, Genomes, Genetics*. 2015;5(4):605–614.
9. Scholz SA, Diao R, Wolfe MB, Fivenson EM, Lin XN, Freddolino PL. High-resolution mapping of the Escherichia coli chromosome reveals positions of high and low transcription. *Cell systems*. 2019;8(3):212–225.
10. Kim S, Beltran B, Irnov I, Jacobs-Wagner C. RNA polymerases display collaborative and antagonistic group behaviors over long distances through DNA supercoiling. Available at SSRN 3263012. 2018;.
11. Balbontín R, Figueroa-Bossi N, Casadesús J, Bossi L. Insertion hot spot for horizontally acquired DNA within a bidirectional small-RNA locus in Salmonella enterica. *Journal of bacteriology*. 2008;190(11):4075–4078.
12. Boyd EF, Almagro-Moreno S, Parent MA. Genomic islands are dynamic, ancient integrative elements in bacterial evolution. *Trends in microbiology*. 2009;17(2):47–53.
13. Touchon M, Hoede C, Tenaillon O, Barbe V, Baeriswyl S, Bidet P, et al. Organised genome dynamics in the Escherichia coli species results in highly diverse adaptive paths. *PLoS genetics*. 2009;5(1):e1000344.
14. Glover N, Dessimoz C, Ebersberger I, Forslund SK, Gabaldón T, Huerta-Cepas J, et al. Advances and Applications in the Quest for Orthologs. *Molecular biology and evolution*. 2019;36(10):2157–2164.
15. Schmid MB, Roth JR. Selection and endpoint distribution of bacterial inversion mutations. *Genetics*. 1983;105(3):539–557.
16. Seemann T. Prokka: rapid prokaryotic genome annotation. *Bioinformatics*. 2014;30(14):2068–2069.
17. Emms DM, Kelly S. OrthoFinder: solving fundamental biases in whole genome comparisons dramatically improves orthogroup inference accuracy. *Genome biology*. 2015;16(1):157.
18. Treangen TJ, Ondov BD, Koren S, Phillippy AM. The Harvest suite for rapid core-genome alignment and visualization of thousands of intraspecific microbial genomes. *Genome biology*. 2014;15(11):524.
19. Schliep KP. phangorn: phylogenetic analysis in R. *Bioinformatics*. 2010;27(4):592–593.

20. Kurtz S, Phillippy A, Delcher AL, Smoot M, Shumway M, Antonescu C, et al. Versatile and open software for comparing large genomes. *Genome biology*. 2004;5(2):R12.
21. Lawrence M, Huber W, Pages H, Aboyoun P, Carlson M, Gentleman R, et al. Software for computing and annotating genomic ranges. *PLoS computational biology*. 2013;9(8):e1003118.
22. Darling AC, Mau B, Blattner FR, Perna NT. Mauve: multiple alignment of conserved genomic sequence with rearrangements. *Genome research*. 2004;14(7):1394–1403.
23. Arndt D, Grant JR, Marcu A, Sajed T, Pon A, Liang Y, et al. PHASTER: a better, faster version of the PHAST phage search tool. *Nucleic acids research*. 2016;44(W1):W16–W21.
24. Shannon P, Markiel A, Ozier O, Baliga NS, Wang JT, Ramage D, et al. Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome research*. 2003;13(11):2498–2504.
25. Clarke BR, Pearce R, Roberts IS. Genetic organization of the *Escherichia coli* K10 capsule gene cluster: identification and characterization of two conserved regions in group III capsule gene clusters encoding polysaccharide transport functions. *Journal of bacteriology*. 1999;181(7):2279–2285.
26. Taylor CM, Roberts IS. Capsular polysaccharides and their role in virulence. In: *Concepts in Bacterial Virulence*. vol. 12. Karger Publishers; 2005. p. 55–66.
27. Lukáčová M, Barak I, Kazar J. Role of structural variations of polysaccharide antigens in the pathogenicity of Gram-negative bacteria. *Clinical microbiology and infection*. 2008;14(3):200–206.
28. Cress BF, Englaender JA, He W, Kasper D, Linhardt RJ, Koffas MA. Masquerading microbial pathogens: capsular polysaccharides mimic host-tissue molecules. *FEMS microbiology reviews*. 2014;38(4):660–697.
29. Dogan B, Suzuki H, Herlekar D, Sartor RB, Campbell BJ, Roberts CL, et al. Inflammation-associated adherent-invasive *Escherichia coli* are enriched in pathways for use of propanediol and iron and M-cell translocation. *Inflammatory bowel diseases*. 2014;20(11):1919–1932.
30. Viladomiu M, Kivolowitz C, Abdulhamid A, Dogan B, Victorio D, Castellanos JG, et al. IgA-coated *E. coli* enriched in Crohn's disease spondyloarthritis promote TH17-dependent inflammation. *Science translational medicine*. 2017;9(376):eaaf9655.
31. Rakitina DV, Manolov AI, Kanygina AV, Garushyants SK, Baikova JP, Alexeev DG, et al. Genome analysis of *E. coli* isolated from Crohn's disease patients. *BMC genomics*. 2017;18(1):544.
32. Oliveira PH, Touchon M, Cury J, Rocha EP. The chromosomal organization of horizontal gene transfer in bacteria. *Nature communications*. 2017;8(1):841.
33. Rowe-Magnus DA, Guerout AM, Ploncard P, Dychinco B, Davies J, Mazel D. The evolutionary history of chromosomal super-integrins provides an ancestry for multiresistant integrins. *Proceedings of the National Academy of Sciences*. 2001;98(2):652–657.

34. Boudeau J, Glasser AL, Masseret E, Joly B, Darfeuille-Michaud A. Invasive ability of an *Escherichia coli* strain isolated from the ileal mucosa of a patient with Crohn's disease. *Infection and immunity*. 1999;67(9):4499–4509.
35. Clermont O, Christenson JK, Denamur E, Gordon DM. The *Clermont* *Escherichia coli* phylo-typing method revisited: improvement of specificity and detection of new phylo-groups. *Environmental microbiology reports*. 2013;5(1):58–65.
36. Alikhan NF, Petty NK, Zakour NLB, Beatson SA. BLAST Ring Image Generator (BRIG): simple prokaryote genome comparisons. *BMC genomics*. 2011;12(1):402.
37. Alekseyev MA, Pevzner PA. Breakpoint graphs and ancestral genome reconstructions. *Genome research*. 2009;19(5):943–957.
38. Rakocevic G, Semenyuk V, Lee WP, Spencer J, Browning J, Johnson IJ, et al. Fast and accurate genomic analyses using genome graphs. *Nature Publishing Group*; 2019.