Genome Complexity Browser: Visualization and quantification of genome variability

Alexander Manolov1Y\*, Dmitry Konanov1Y, Dmitry Fedorov1, Ivan Osmolovsky1, Rinat …, Elena Ilina1

**1** Federal Research and Clinical Centre of Physical and Chemical Medicine, Federal

Medical and Biological Agency of Russia, Moscow, Russian Federation

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

# Abstract

Comparative genomics studies may be used to acquire new knowledge regarding genome architecture, which defines the rules for combining sets of genes in the genome of living organisms. Hundreds of thousands of prokaryotic genomes have been sequenced and assembled. However, computational tools capable of simultaneously comparing large numbers of genomes are lacking. We developed the Genome Complexity Browser, a tool that allows the visualization of gene contexts, in a graph-based format, and the quantification of variability for different segments of a genome. The graph-based visualization allows the inspection of changes in gene contents and neighborhoods across hundreds of genomes, simultaneously, which may facilitate the identification of conserved and variable segments of operons or the estimation of the overall variability associated with a particular genome locus. We introduced a measure called complexity, to quantify genome variability. Intraspecies and interspecies comparisons revealed that regions with high complexity values tended to be located in areas that are conserved across different strains and species.

# Author summary

The comparison of genomes among different bacteria and archaea species has revealed that many species frequently exchange genes. Occasionally, such horizontal gene transfer events result in the acquisition of pathogenic properties or antibiotic resistance in the recipient organism. Previously, the probabilities of gene insertions were found to vary, with unequal distributions along a chromosome. At some loci, referred to as hotspots, changes occur with much higher frequencies compared with other regions of the chromosome. We developed a computational method and a software tool, called Genome Complexity Browser, that allows the identification of genome variability hotspots and the visualization of changes. We compared the localization of various hotspots and revealed that some demonstrate conserved localizations, even across species, whereas others are transient. Our tool allows users to visually inspect the patterns of gene changes in graph-based format, which presents the visualization in a format that is both compact and informative.

# Introduction

The genome is not simply the storage of gene sequences. Many sequence motifs and sequence patterns must be located at proper loci to ensure proper interactions between the molecular machines involved in basic cell processes (transcription, genome replication, and cell division) and the chromosome [1-3]. The interactions between these cellular processes and DNA molecules and the interplay among them govern optimal gene localizations and orientations [1]. New knowledge regarding the genome architecture, which represents the set of imposed constraints and favorable configurations for genomic objects [1], may deepen our knowledge of the basic cellular processes and facilitate the development of artificial genomes in the field of synthetic biology.

The non-random localization of different genes may be important, due to several factors.

Genes located near the replication origin may have higher copy numbers in fast-dividing cells, which is known as the replication-associated gene-dosage effect [4,5]. Chromosomal folding can bring genes located in different regions of the chromosome close to each other in 3D space, which can be beneficial for genes that encode a regulator and its targets genes [6,7]. The gene expression effects of global regulators, such as the histone-like nucleoid-structuring protein (H-NS), have been shown to depend on the location of the target genes [8], and the transcriptional propensity also varies, depending on the position of the gene within the chromosome [9]. The cooperative effects of RNA polymerases [10] and supercoiling propagation may play roles in the transcriptional regulation of neighboring genes.

Horizontal gene transfer (HGT) events are preferentially localized in hotspots, which are chromosomal loci in which changes are observed much more frequently than in other regions [11–13]. Although disruptions in genome architecture may result in the decreased fitness of an organism, changes can be introduced in some regions of the chromosome without inducing negative effects. To our knowledge, no currently available computational tool is capable of performing quantitative estimation of variability along the chromosome. Such a tool is necessary to deepen our knowledge of the factors determining genome variability and stability.

Here, we present the Genome Complexity Browser (GCB), a tool that allows the estimation of local genome variability and visualization of gene rearrangements. Both tasks are performed using graph-based representations of gene neighborhoods within a set of genomes. Local genome variability is evaluated using the metric introduced here, which we referred to as complexity. Complexity profiles may be used to identify hotspots of horizontal gene transfer or other local gene rearrangement events. The graph-based visualization available in GCB allows the analysis of patterns among genome changes events and the detection of persistent or variable gene combinations (e.g., variable and conservative regions of operons).

# Materials and methods

## Graph construction

The input for this step is the set of genomes, with inferred orthogroups. The algorithm for graph construction is the following: each orthogroup 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 and B). Graph objects and their methods are implemented in the gene-graph-lib library for Python 3, and more information can be found within the library documentation at [https://github.com/DNKonanov/gene](https://github.com/DNKonanov/gene_graph_lib)\_[graph](https://github.com/DNKonanov/gene_graph_lib)\_[lib.](https://github.com/DNKonanov/gene_graph_lib)

**Fig 1. Principal scheme used for the graph-based representation of gene order in a set of genomes and the genome variability estimation approach.** To construct a graph, each orthogroup 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, and 3 represent three different hypothetical genomes. The arrows represent genes, and genes within the same orthogroup have the same color and letter designation. B. Graph-based representation of the 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.

Because GCB uses directed graph-based representations of gene order, all genomes in a set are first coaligned, to achieve the same orientation throughout the set. This step is performed automatically, and can be optionally skipped because it takes a lot of time with a large number of draft genomes (runtimes are shown in the results section). The algorithm used for this step can be found in S1 Listing.

Orthogroups may include paralog genes [14], and some modifications to the basic process used for graph construction are necessary in this case. Two methods are suggested in GCB. The first method does not include the orthogroup in the graph for genomes containing paralogs (however, the same orthogroup will be included in the graph for genomes containing no paralogs), shown in S1 Fig A. Thus, if some orthogroup contains two paralogs from only one genome, then the orthogroup will be excluded from graph for this genome, while still being included for all other genomes. This approach has the advantages of simplicity and clear output, although some genes will be missed in the graph. The second approach is to “orthologize” paralog genes: for each set of paralogous genes with a unique context, a graph node with a unique suffix will be created (S1 Fig B). GCB uses the first approach by default, whereas the second approach is available as an option, in both the command line and browser-based versions.

## Genome complexity definition

We introduce a measure called complexity for the quantification of local genome variability. Complexity values are calculated against one reference genome in the set. This genome is extracted from the graph as a simple chain of nodes, called the reference chain (Fig 1C). To calculate complexity in node X, nodes from the reference chain, in the range ±*window,* around node X are selected, and the complexity is defined as the number of distinct paths in the graph that do not contain the node X but that start and finish in the nodes from the selected range (deviating paths), as shown in 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 the complexity values of all nodes in the reference between the nodes that represent the start and end of the deviating path (Algorithm2).

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

**Hotspot definition**Genome complexity profiles often contains peaks that are surrounded by regions with relatively low values. This complexity peaks correspond to the regions with high local variability, called variability hotspots. Genes are considered to belong to a hot-spot if their complexity exceeds a threshold, defined as the third quartile value plus the interquartile range, multiplied by the coefficient, *k* (*k* equals 1.5, by default), which is an arbitrary but commonly used criterion for outlier detection, initially proposed by Tukey [Tukey, 1977]. The coefficient, *k,* can be altered by the user, to obtain the only highest values or to include modestly complex regions. Hotspot region coordinates can be downloaded in the web-based version and can be obtained in the command line version. Because no rigid mathematical definition for hotspots exists, users can infer them with their preferred methods and thresholds, after downloading the complexity values from a GCB web site or calculating them using the stand-alone, command-line version.

**Simulations of genomes with predefined variability profiles.**

We hypothesized that the calculated complexity values observed for some regions of a chromosome correlated with the frequencies of fixed rearrangements in that region. To verify this assumption and to 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 set as equal to the deletion event probability, to maintain the genome length. The probability of inversions was set to 1/100 the number of other events, based on data in the literature reporting that inversion events are less common than other types of rearrangements, such as deletions and duplication [15]. The localization of these changes throughout the chromosome was determined according to the predefined variability profiles. 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 can be found in the 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* ← empty set *i* ← 1

**while** *i* ≤ iterations **do** *next node* ← select random node connected with start node *path* ← [ start node ] **while** *next node* **not in** ref chain **do**

**if** *next node* **not in** *path* **do**

add *next node* to *path*

*next node* ← select random node connected with the last node in the *path* **if** all nodes connected with *next node* are in *path* **do**

*path* ← [ ] **break**

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

add *path* to *Paths*

*i* ← *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* ← reference nodes chain set initial complexity values for all nodes as 0 **for each** node **in** *ref chain* **do**

*Paths* ← FIND PATHS(*graph, node, ref chain, number of iterations*) **for each** path **in** *Paths* **do** *start* ← first node in the path *end* ← last node in the path

*distance* ← |position(*start*) - position(*end*)| **if** *distance* ≤ window **do**

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

*complexity*[*node*] ← complexity[node] + 1/window

**return** complexity values

## Subgraph generation

To visualize a gene context in the region of interest, a subgraph representing this region can be 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 in the set and adds deviating paths that are limited to the selected region. If the length of the path is greater than the *depth* parameter, then the path is cropped, and only the start and end fragments (tails) of a fixed length (*tails* parameter, *tails <depth*) are added to the subgraph. If the weight of any edge is less than the user-defined *minimal edge weight* parameter, this edge is not added to the subgraph. The subgraph generation algorithm can be found in the S3 Listing.

## Web server data acquisition and preparation

To construct a dataset for the web server, we downloaded genomes for 143 prokaryotic species with more than 50 genomes available in the RefSeq database. For each species, if the number of complete available genomes 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 the *Escherichia coli* extended genome set, which contained 327 complete genomes, as of November 2017.

All downloaded genomes were reannotated with Prokka ver 1.11 [16] to achieve uniformity. Genes were assigned to orthogroups with OrthoFinder ver. 2.2.6 [17]. Python scripts contained within the GCB application were used to parse OrthoFinder outputs, 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 the RI function from R phangorn library [19]. To estimate the similarity to the reference genome, as in the analysis presented in Figure 6, all genomes were aligned with nucmer [20], and a similarity score was calculated as follows: all aligned reference genome ranges were reduced with IRanges R package [21], their total lengths were divided by the reference genome length, and all query genomes were sorted by this value, after which the strains with the 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 the following parameters: *tails* = 1 and *minimaledge* = 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 and *minimaledge* = 5. The code used to generate Figures 4-6 is available at [https://github.com/paraslonic/GCBPaperCode.](https://github.com/paraslonic/GCBPaperCode)

# Results

## Software description and availability

The GCB tool is available as both a standalone application and a web server. The GCB web server is located at http://gcb.rcpcm.org and contains data for 143 prokaryote species. A subset of the available genomes was included in analyses for which the number of available genomes was greater than 100 (except for E. coli). The complexity profiles in the web version were calculated with window sizes of 20, 50, and 100 genes. If a user desires to perform an analysis on a custom set of genomes or with different window sizes, then the standalone version should be used. All features of the web version are also available in the standalone version. To use the standalone version, a user should have basic command-line program operating skills. No precalculated data is available in the standalone version.

In this section we will first describe graphic user interface (GUI) elements, followed by the use of the standalone version.

**Elements of graphic user interface**

GCB browser-based GUI consists of four primary components: 1) the top panel, which allows the selection of a genome and a region; 2) a search panel; 3) the complexity plot; 4) the subgraph visualization (Figure 2).

**Fig 2. Screenshot of the GCB browser-based interface**. The GCB GUI is available at gcb.rcpcm.org and when GCB is run as a local server. The GUI consists of a panel in which the user can select an organism, a complexity profile panel, and a subgraph visualization panel, with a box containing information regarding nodes and edges.

In the *top panel,* the user can select the genome set (one per species in the web server, an arbitrary set in the standalone version), a particular genome, and a contig (in the case of a draft genome or when a genome consists of several replicons). When a genome has been selected, the complexity profile of the selected genome will be plotted in the complexity plot panel. The user can also specify the coordinates for a specific region of interest, which will be visualized in a graph form. The size of the region should be limited between dozens and a few hundred kilobases (depending on the region complexity), to enable the performance of the graph visualization step.

The user can search gene annotations to identify the locations of genes of interest, using the *search panel*. Searching is performed over the product features of annotated genomes (only protein-coding sequences are considered).

The *complexity plot panel* shows a visualization of the complexity profile for the selected genome. Numeric values associated with the complexity profile can be downloaded as a text file, for further analysis (e.g. comparisons against other profiles for other organisms). The visualization of custom data can also be added (such as GC content, pathogenicity islands, prophage regions, and sequence motifs) by supplying a file, formatted as:  <genome position> <numeric value>.

The *graph visualization panel* shows a graph-based representation of a selected region of the genome. Several settings are available for customizing the subgraph visualization, to simplify the analysis, (e.g., by setting the minimal depicted edge weight). Subgraphs can be saved as JPEG images or exported in JSON format, which can be visualized using specialized software (e.g., Cytoscape) for the preparation of publication-ready images.

**Standalone version**

For the analysis of a custom set of genomes, we have provided a set of scripts that will enable users to perform the following tasks: 1) infer orthogroups, 2) generate a graph, 3) estimate complexity values in text form and an SQLite database file, 4) generate and visualize subgraph. Database files can be imported to a local GCB server, which can be run on a standard PC. Fig 2B shows a roadmap for the standalone analysis.

Fasta-formatted genomes are expected as the inputs (both complete and draft assemblies may be used, although, we recommend the inclusion of at least one complete genome to be used as a reference). Snakemake script (<https://github.com/paraslonic/orthosnake>) is provided to infer orthogroups. It performs genome annotation with Prokka [], generates protein sequences in fasta format, with position and product information in the header, and makes orthogroup inferences with OrthoFinder [].

Python scripts, which are available at <https://github.com/DNKonanov/geneGraph>, should be used for further analysis steps. The OrthoFinder output serves as the input for the *parse\_og.py* script, which produces a database and a sif-formatted file containing the graph-based representation of all of the genomes. The Python3 script *estimate\_complexity.py* computes the complexity profile for the specified reference genome. Optionally, this analysis step can be limited to subset of genomes by providing text files containing genome file names, which can be useful for excluding some genomes after orthogroups are inferred or for analyzing the variability of a specific subset of genomes). Tutorials for performing these analyses are available at ….

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 orthogroups, which is the most computationally difficult step.

## Subgraph visualization

Graph-based representations of gene order can provide convenient methods for visually inspecting the contexts of genes of interest and to identify conservative and variable gene combinations. GCB can construct and visualize subgraphs, which are those parts of the genome graph that contain the region of interest. Next, we will describe examples of subgraphs generated using GCB and visualized with Cytoscape [24].

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

Fig 3A shows a subgraph representing the gene context of the capsule gene cluster (chromosomal coordinates 3111444-3128026 in NCBI sequence NC 011993.1) for 327 complete *Escherichia coli* genomes. From this visualization, the operon can be observed to contain two conserved regions and one variable region. The variable region consists of genes from the serotype-specific synthesis region, whereas the neighboring conserved regions correspond with regions involved in polysaccharide export [25]. The capsule is considered to be an important virulence factor [26] for *E. coli* and many other bacterial species, and capsule variations are essential for the avoidance of immune responses and phage infections [27,28].

Although the existence of variable and conserved regions of this operon was previously known, new information regarding the architecture of other operons may be obtained through analyses performed in GCB.

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

Fig 3 B and C show the visualization of subgraphs for regions containing two operons: hemin uptake (hmu) and propanediol utilization (pdu). The presence of *E. coli* harboring these operons in the intestinal microbiome has previously been associated with Crohn’s disease [29–31]. These operons have different phylogenetic

distributions, with the hmu operon being preferentially present in the B2 phylogroup (S2 Fig A, retention index = 1), and the pdu operon can being found in phylogenetically distinct strains of *E. coli*, where its presence is in low agreement with the phylogenetic tree (S2 Fig B, retention index = 0.26). The hmu operon is located at the 3691615-3700567 positions and the pdu operon is located at the 2083448-2101340 positions of the NCBI Reference Sequence NC 011993.1.

**Sthe ativeoperon regionscan in**forcontaining:the G-basededregions the hemin uptake and propanediol utilization edregionssregionseveral genomes, where is located,can be found

The graph-based visualization reveals that the hmu operon is located in a conserved region, in which the neighboring genes are the same in all strains in which this operon can be found (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-based visualization also indicates that one of the genes (hemin transport system permease, HmuU) or close homologs are present in two alternative contexts.

The pdu operon is harbored by only a fraction of all considered strains (27 out of 327), but is also located in a conserved region (Fig 3 C). Some variations in the pdu operon are visible and reflect different operon variants [31]. Unlike the hmu locus, here alternative gene sets are present. These alternative sets include genes associated with iron transport (FepC, FcuA, and HmuU), DNA mobilization (retroviral integrase core domain and the transposase DDE Tnp ISL3). These alternative sets can have high variability, with many overlapping changes observed in the subgraph visualization.

In the next section, we will describe the quantitative measurement of this observable difference in subgraph complexity.

## Complexity is a measure of genome variability

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

We implemented an algorithm (Algorithm 2, described in the Methods) to count the number of distinct random walks in a subgraph that represents a given region of the reference genome, and the computed value is referred to as region complexity. By selecting subregions with a sliding window and calculating the complexity value for these subregions, we can obtain the complexity profile of the reference genome. The size of the sliding window can be set by the user, and the default value of 20 was used for the results described below.

**Method verification**

We verified our approach by performing genome evolution simulations based on potential gene transfer events, by comparing with previously published results and through the analysis of known variability hotspot regions (integrons).

The complexity profile for *E. coli*, ascalculated by GCB, is in good agreement with the hotspots evaluated in [32] (Fig 4A). Complexity values within the hotspots identified in [32] were significantly greater than those outside of the hotspots (p-value *<*10−16, Mann-Whitney U test, see Fig 4B).

**Fig 4. Complexity values can be used as genome variability measures.** A) Comparisons of complexity profiles for *E. coli K12,* with hotspots identified in [32] (blue rectangles underneath complexity profiles). B) Comparisons of complexity values among genes located inside and outside of the hotspots identified in [32]. C) Comparisons between the initial variability profiles and the complexity profiles that were calculated based on the artificial genomes, after 3,000 evolution simulation steps. D) Complexity profiles for *Vibrio cholerae N16961,* chromosome II, with noticeably high levels of complexity at the integron regions.

We performed several simulations, during which we suggested that the probability of genomic rearrangement events (HGTs, deletions, and translocations) was 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.

Integron and they have high complexity values (see Fig )

Integrons are gene acquisition systems capable of integration, excision and rearrangement of gene cassettes and are examples of genome variability hotspots. We observe that integron regions have high complexity values. Fig 3D shows integron region of *V. cholera* as an example. This integron 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 but not limited to them**

Fig 5 shows the complexity profile along the chromosome of adherent-invasive *E. coli LF82* [34]. As expected, the regions with a high 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 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) 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**

Genome complexity analysis can be used to compare variability profiles of different species or intraspecies structures (e.g. phylogroups). We performed comparison of the complexity profiles for 146 species and observed that when genomes are similar enough (synteny blocks covers most part of the genomes), then complexity profiles are also similar (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. 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 (e.g. 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, green triangle denotes conserved region with high variability not associated with prophages and genome islands; 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.  
 We described above a region in *E. coli* genome with high variability rate and without identifiable mobile genetic elements (designated with green triangle in Fig 6C). As can be seen from Fig6C this variability hotspot is present in A, B1, B2, D and to lesser extent in E phylogroup. Phylogroup E consisted of genomes closely related to *O157:H7 Sakai* strain and contains the largest genomes, mainly due to expansion of bacteriophages. We observed that only in this phylogroup this region has integrated prophage. Prophage integration can explain variability in E phylogoup, but what is a driving force of high variability of the region in other phylogoups remain to be elucidated.

**Method applicability**

Complexity profiles and subgraphs could be obtained for any set of genomes for which orthogroups could be inferred. Graph representation, window-based variability estimation and subgraph visualization performs best when the set contains closely related genomes in which local variability does not overwhelmed by chromosomal rearrangements longer then the chosen window size. From our experience and estimates from other studies [Brilli, BMC Genomics 2013] this means that genomes within 0.05 phylogenetic distance (approximately, a species boundary [Mash 2016]) should be used. Complexity profiles of different sets of genomes can be compared with same limitations, large amounts of genome rearrangements make comparison not informative. We observed that some species have no obvious peaks in the complexity profile (for example naturally competent *P. fluorescens*), which makes comparative analysis not as informative as for species with clear regions of low and high complexity, see SFig XA. Meanwhile naturally competent *N. gonorrhoeae* gives quite clear and comparable profiles (SFig XB).

Draft genomes (consisting of fragmented genome regions called contigs) may be used for complexity estimation. We performed comparisons of complexity profiles inferred with 100 complete or 100 draft genomes with same complete genome as reference and observed significant similarity with pearson correlation coefficient equals 0.87 (Fig 7). The code for the analysis is available at github. The impact of draft genomes may be higher in the case of highly stable genomes with large scale rearrangements as main source of variability (e.g. Mycobacterium tuberculosis). Subgraph visualization suffers from genome fragmentation because false negatives may be introduced by contig boundaries (for example no context of region representing some particular contig could be identified).

**Fig 7 Method applicability and benchmark.** A) complexity profiles computed with 100 complete genomes (at the top) or 100 draft genomes (at the bottom) of *E. coli* are very similar, B) Correlation of complexity values obtained with either 100 *E. coli* genomes or subsets with lower number of genomes. C) Time needed for the graph construction and complexity estimation based on different number of genome.

Small number of genomes included in the analysis may lower the accuracy of complexity profile estimation. We recommend to use not less than several dozens. Fig 7 shows the correlation of complexity values obtained with either 100 *E. coli* genomes or subsets with lower number of genomes. When more than 40 genomes are included in the analysis, pearson correlation coefficient becomes greater than 0.9.

Time needed for the analysis depends on the number of genomes. Fig 7 shows time needed for the graph construction and complexity evaluation steps for a different number of genomes (up to 1000). Main time consuming step for the overall analysis is the orthogroup inference. When no computational cluster is available and the number of genomes is large, other method than orthofinder may be considered. In this case users will be needed to format orthogroup information as: <orthology\_gourp\_id>: gene\_id1 gene\_id2 …, with one line per orthogroup (orthofinder output format).

Local variability hotspots identified with GCB are often coincide with prophages and genomic islands. We compared identified hotspots to curated literature-based dataset [Bertelli et al., 2019] and obtained 0.65 mean F1 score (which is comparable to existing tools such as IslandPath-DIMOB, GIHunter and IslandViewer 4), and 0.54 mean precision. Comparison to the automatically generated dataset gives much lower accuracy scores, see S2 Text for detailed information. We conclude that complexity analysis should be considered as a helper in explorative analysis of genomic islands or prophages, but should not be used as a genomic island predictions without additional analysis. This is because hotspots may be of a different (often unknown) origin.

# Discussion

Synteny visualization tools (e.g. 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 synteny 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. Still, paralogous genes may be attributed to one orthogroup which makes graph representation of the context problematic. We observed that on the average 0.5% of all orthogroups per genome contains at least one paralogues gene; but among all orthogroups inferred for the species the proportion of orthogroups with paralogs is almost 16% (see S1 Table for information for each species individually, code is available at <https://github.com/paraslonic/GCBPaperCode/tree/master/GeneCount>). 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.

1. 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;.
2. Balbont´ın R, Figueroa-Bossi N, Casadesu´s 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.
3. 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.
4. 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.
5. Glover N, Dessimoz C, Ebersberger I, Forslund SK, Gabaldo´n T, Huerta-Cepas J, et al. Advances and Applications in the Quest for Orthologs. Molecular biology and evolution. 2019;36(10):2157–2164.
6. Schmid MB, Roth JR. Selection and endpoint distribution of bacterial inversion mutations. Genetics. 1983;105(3):539–557.
7. Seemann T. Prokka: rapid prokaryotic genome annotation. Bioinformatics. 2014;30(14):2068–2069.
8. Emms DM, Kelly S. OrthoFinder: solving fundamental biases in whole genome comparisons dramatically improves orthogroup inference accuracy. Genome biology. 2015;16(1):157.
9. 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.
10. Schliep KP. phangorn: phylogenetic analysis in R. Bioinformatics.

2010;27(4):592–593.

1. 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.

1. 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.
2. Darling AC, Mau B, Blattner FR, Perna NT. Mauve: multiple alignment of conserved genomic sequence with rearrangements. Genome research. 2004;14(7):1394–1403.
3. 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.
4. 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.
5. 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.
6. Taylor CM, Roberts IS. Capsular polysaccharides and their role in virulence. In: Concepts in Bacterial Virulence. vol. 12. Karger Publishers; 2005. p. 55–66.
7. Luk´aˇcov´a 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.
8. 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.

1. 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.
2. 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.
3. 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.
4. Oliveira PH, Touchon M, Cury J, Rocha EP. The chromosomal organization of horizontal gene transfer in bacteria. Nature communications. 2017;8(1):841.
5. Rowe-Magnus DA, Guerout AM, Ploncard P, Dychinco B, Davies J, Mazel D. The evolutionary history of chromosomal super-integrons provides an ancestry for multiresistant integrons. Proceedings of the National Academy of Sciences. 2001;98(2):652–657.
6. Boudeau J, Glasser AL, Masseret E, Joly B, Darfeuille-Michaud A. Invasive ability of an Escherichia colistrain isolated from the ileal mucosa of a patient with Crohn’s disease. Infection and immunity. 1999;67(9):4499–4509.
7. Clermont O, Christenson JK, Denamur E, Gordon DM. The C lermont E

scherichia coli phylo-typing method revisited: improvement of specificity and detection of new phylo-groups. Environmental microbiology reports. 2013;5(1):58–65.

1. Alikhan NF, Petty NK, Zakour NLB, Beatson SA. BLAST Ring Image Generator (BRIG): simple prokaryote genome comparisons. BMC genomics. 2011;12(1):402.
2. Alekseyev MA, Pevzner PA. Breakpoint graphs and ancestral genome reconstructions. Genome research. 2009;19(5):943–957.
3. 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.