Computational Genome Analysis Using The G-language System

Kazuharu Arakawa§ • Haruo Suzuki • Masaru Tomita

1 Institute for Advanced Biosciences, Keio University, Fujisawa 252-8520, Japan
First two authors are equal contributors
Corresponding author: gaou@sfc.keio.ac.jp

ABSTRACT

Computational genome analysis requires sophisticated workflows, seamlessly uniting multiple tools and algorithms. In order to maximize the productivity of genomics research with bioinformatics, a computational framework that allows rapid integration of available resources is desirable. G-language Genome Analysis Environment is a generic open-source workbench for this purpose, with the aim to: 1) construct an integrated analysis and development environment for bioinformatics, 2) systematically accumulate and implement existing algorithms and data, and to 3) aid the construction of analysis workflows. This system provides over 200 analysis methods for genome informatics and systems biology, with programmable interfaces, an interactive command-line shell, and a graphical user interface. Here we review the methods and algorithms implemented in this system especially focusing on genome informatics analysis, including methods for the identification of sequences with significant information content using information theory, observation of nucleotide composition and genomic compositional asymmetry, calculation of codon bias measures and prediction of gene expression levels, and statistical analysis of short oligomers such as short tandem repeats and palindromes. Since these methods are combined with several other applications and algorithms to produce a workflow in genome informatics research for studying specific biological questions, we also present brief overviews of workflows utilizing these algorithms used in several genomic studies.

Keywords: bioinformatics, codon, G-language Genome Analysis Environment, genome informatics, GC skew, genomics
Abbreviation: PWM, position weight matrix

INTRODUCTION

Molecular biology has quickly become a data-driven science. Bioinformatics is now an indispensable means in order to cope with the ‘explosion’ of data, exemplified by the hundreds of completely sequenced genomes and evermore-increasing omics information produced by high-throughput experiments (Butler 2001; Arakawa et al. 2006b; Lioios et al. 2006). Computational biology has initially evolved synergistically with the genome projects, especially in sequence assembly, gene identification, and annotation of the genomic information, greatly contributing to the success of the Human Genome Project (Stein 1996; Chiczurel 2002; Collins et al. 2003; Hood et al. 2003). At this stage the primary role of bioinformatics was data processing and generation, but with the progress in genomics and the advent of systems biology, it is rapidly expanding into the fields of knowledge discovery through data integration and mining of the masses of information, and hypothesis generation and testing (Kitano 2002a, 2002b; Kell et al. 2004; Ideker et al. 2006). Central to the advent of this new paradigm is the availability of effective software infrastructures. A myriad of bioinformatics tools have been developed for specific analyses, including de facto standard sequence analysis
software tools such as BLAST (Altschul et al. 1997), HMMER (Eddy 1998), and ClustalW (Thompson et al. 1994).

Since computational biology encompasses extremely broad areas of molecular biology that are at the same time rapidly expanding and evolving, development of any one feature-rich and versatile application for bioinformatics would likely not work to suit the diversity of research fields. Instead, specialized software tools and components that are developed to perform specific analyses can be dynamically linked together, creating a workflow to achieve the intended research depth (Swertz et al. 2007). Development of tailor-made software by combining functional units to meet diverse needs using the command-line environment of UNIX operating systems has been shown to be efficient, because the programs can be “piped” to perform complex functions. For example, the combination of merely 10 command-line programs is possible in theory to generate $10! = 3,628,800$ variations, which requires much less development cost than one multi-functional application with 100 features. A similar design principle can be seen in the organization of the human genome, where extremely diverse biological functions arise from a limited number of genes. This approach, called “mash-up”, is actively utilized in the current developments of web-applications as a cost-effective means to create diverse products (Belleau et al. 2008). In order to connect different software tools in a pipeline, at least some computer programming is essential to filter the inputs and outputs, and a software infrastructure that allows minimal programming and easy connection of the components is necessary for effective computational molecular biology.

Current efforts for such bioinformatics software infrastructures mainly focus on one of the three interfaces of the computational environment: application programming interface (API), command-line user interface (CUI), and graphical user interface (GUI). BioPerl, BioPython, BioJava, and BioRuby, collectively known as the Bio* toolkits (Mangalam 2002; Stajich et al. 2002), provide APIs for easy handling of the various biological databases and software tools in corresponding programming languages. With these toolkits, bioinformatics developers can access the data as native objects without worrying about the differences in data formats, and at the same time take advantage of the text

Fig. 1 Graphical User Interface (GUI) of G-language GAE. Viewing clockwise from top left corner, shown windows are 1) main control panel, 2) console, 3) text output window, 4) result from view_cds program showing nucleotide contents around start/stop codons, 5) result from genomicskew program showing GC skew of multiple regions of genome, 6) result from genome_map program showing gene locations and nucleotide contents, and 7) configuration window for the manipulation of workflow. Using the GUI, users can run the programs implemented in G-language GAE in a workflow without writing a single line of code.
string processing abilities of the scripting languages that are especially convenient for biological sequence information. The Bioconductor project (Gentleman et al. 2004) also provides APIs for bioinformatics especially focusing on gene expression analyses, data visualization, and statistical analyses, taking advantage of the rich features of R statistics language, which is available at http://www.r-project.org/. Since the R language is equipped with an interactive shell environment, the user experience of Bioconductor is similar to that of CUI. EMBOSS (European Molecular Biology Open Source Software Suite) (Rice et al. 2000) and NCBI SEALS (System for Easy Analysis of Lots of Sequences) (Walker et al. 1997) provide large collections of command-line applications for data retrieval and manipulation, sequence analyses, phylogenetic analyses, and numerous others that can be linked into workflows. The majority of bioinformatics software is distributed in the form of command-line applications, and therefore these toolkits can work seamlessly together. Moreover, these command-line tools are also often provided as web-services based on Simple Object Access Protocol (SOAP) or Representational State Transfer (REST) with BioMOBY standards (Wilkinson et al. 2005), and Taverna workbench provides a GUI to utilize these web-services and to formulate workflows (Oinn et al. 2004). It is worth noting that these projects are mostly based on open-source development for high accessibility, dynamism, and transparency that is required for scientific purposes in order to be able to examine the methods and algorithms employed within the software tools. All of these interfaces are practical solutions depending on the use cases, due to the tradeoffs between scalability and accessibility. Programming using APIs is the most scalable interface, but it is also the least accessible one since it requires certain level of programming knowledge and skills. Conversely, GUI is most accessible with the intuitive interface that can be manipulated with a mouse, but applications of this type are limited in terms of scalability. Multiple interfaces are sometimes employed to improve the flexibility of the software. For example, EMBOSS also provides comprehensive development APIs and a graphical front-end named Jemboss (Carver et al. 2003), and BioPerl includes interfaces for EMBOSS. SOAP-based web-services can be accessed through APIs, and Taverna also allows scripting by Java within the GUI application for customized filtering of the data.

Bioinformatics currently belongs to the molecular biology domain, and therefore genome sequence data is central to this discipline. The digital nature of biological sequence information makes computer programming a highly suitable means for the manipulation of this data (Hood et al. 2003), especially for scripting languages such as Perl that is specialized for text processing. Programming is also necessary for genome informatics research in terms of scalability of software development by mash-ups of existing tools and algorithms. However, interactive and responsive CUI as well as accessible and user-friendly GUI would also facilitate the research processes. In light of these requirements for computational genome informatics, the G-language Project at the Institute for Advanced Biosciences, Keio University, Japan have been developing a generic workbench designated G-language Genome Analysis Environment (G-language GAE) since 2001 (Arakawa et al. 2003). The software system is equipped with interchangeable interfaces as API, CUI, and GUI for high scalability and accessibility, with Perl-based API that is compatible with BioPerl, more than 200 analysis applications especially focusing on genome informatics studies many of which are not available through other software packages, and an intuitive GUI that can be easily converted from Perl scripts (Fig. 1). G-language GAE is developed as an open-source software, distributed under GNU General Public License at http://www.g-language.org/. Currently available version of the software is 1.8.4, but version 2 with enhanced user interface is also under development. Details of the internal architecture of the software system is reviewed elsewhere (Arakawa et al. 2006a).

G-language GAE is especially strong for genome informatics analyses, with numerous algorithms implemented to be directly accessible from Perl programming language (and with BioPerl sequence object). In this work, we review the methods and algorithms implemented in G-language GAE for genome informatics studies, concentrating on the following areas: identification of binding sites based on information theory, analysis of genomic compositional skew, and analysis of synonymous codon usage bias. Combined with other software tools and scripts, utilization of these algorithms should facilitate the development of workflows for computational genome analysis.

IDENTIFICATION OF BINDING SITES WITH INFORMATION THEORY

A typical starting point in sequence analysis is the identification of conserved sequence elements or motifs in order to characterize functional sequence structures, such as transcription factor binding sites and Shine-Dalgarno (Shine et al. 1974)/Kozak (Kozak 1987) sequences for ribosome binding sites. In order to identify such binding sites, multiple sequences are typically aligned to formulate a position weight matrix (PWM) (Stormo et al. 1982), and conservation is thereby quantified mathematically based on Claude Shannon’s information theory (Shannon 1948), which is applied to nucleotide and protein sequences by the works of Schneider and colleagues (Schneider 1997; Schneider 2002). Commonly used indices for this purpose include Shannon uncertainty (entropy) $H$, information content $I$, and Kullback-Leibler divergence. Sequence Logo software (Schneider et al. 1990) and WebLogo online generator (Crooks et al. 2004) are frequently utilized to visualize the information content in a given set of aligned sequence data (Fig. 2). Naïve calling of the most frequent alphabets for the identification of “consensus” sequence can be erroneous, since such method ignores the frequency information and subsequently the degree of sequence conservation in the given alignment. Methods based on information theory avoid this pitfall by measuring the amount of information (or randomness) to identify conserved residues (Schneider 2002).

Fig. 2 Sequence Logo for Shine-Dalgarno sequence in Escherichia coli. Sequence Logo graphically displays the amount of information content at each position, represented by the height of the stacked alphabet. Height of each nucleotide corresponds to its contributing frequency. Here one can clearly see the conservation of ATG start codon, and purine rich Shine-Dalgarno sequence can be found at positions -7 to -12 (4 to 9 in the figure). WebLogo (Crooks et al. 2004) was utilized for visualization.
Table 1 Programs for consensus analysis implemented in G-language GAE.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>base_entropy</td>
<td>Calculates and graphs the sequence conservation in regions around the start/stop codons using Shannon uncertainty (entropy).</td>
</tr>
<tr>
<td>base_information_content</td>
<td>Calculates and graphs the sequence conservation in regions around the start/stop codons using information content.</td>
</tr>
<tr>
<td>base_relative_entropy</td>
<td>Calculates and graphs the sequence conservation in regions around the start/stop codons using Kullback-Leibler divergence (relative entropy).</td>
</tr>
<tr>
<td>consensus_z</td>
<td>Calculates and graphs the sequence conservation in a given array of sequences, and names a consensus using z-score cutoff.</td>
</tr>
</tbody>
</table>

When conserved regions are identified with information theory, PWM for the putative binding site of interest can be used as training set for machine learning methods, for further search of similar conserved regions and motif prediction (Cartharius et al. 2005; Hertzberg et al. 2005). HMMER software based on Hidden Markov Model (HMM) algorithm has been especially popular in genomics and proteomics (Eddy 1998). Other machine learning approaches frequently utilized for the prediction of protein binding sites include artificial neural networks, self-organizing maps, and support vector machines (SVM), that are reviewed elsewhere (Schneider et al. 2004).

In the following paragraphs, we describe three methods based on the information theory for the analysis of sequence conservation: uncertainty, information content, and Kullback-Leibler divergence. In addition, we describe the z-score method for consensus sequence calling by statistical means. All of these methods are implemented in G-language GAE (Table 1).

**Shannon uncertainty and information content**

Uncertainty of information $H$ (also known as entropy) at given position $i$ with distribution $P_i$ is defined as follows:

$$H(P_i) = -\sum_{j \in M} P_{ij} \log_2 P_{ij} \text{ (bits per symbol)}$$

where $M$ is the set of alphabets representing the sequence units (for DNA, \{A, T, G, C\}) and $P_{ij}$ is the frequency of a certain alphabet at position $i$ (Shannon 1948). Unit of $H$ is “bits” when binary logarithm is used, as in the above formula. $H(P_i)$ is zero and maximum when distribution is most biased and therefore the entire information is represented by only one letter of the alphabet. $H(P_i)$ takes the maximum value of $\log_2 |M|$ bits when the distribution is uniform for all alphabets, where $|M|$ is the cardinality of $M$ (4 for DNA, therefore the maximum $H$ is 2).

Information content $I$ is obtained by subtracting $H$ from the maximum uncertainty $\log_2 |M|$:

$$I(P_i) = \log_2 |M| - H(P_i) = \log_2 |M| - \left( -\sum_{j \in M} P_{ij} \log_2 P_{ij} \right)$$

therefore $I(P_i)$ is maximum when the frequency is most biased to certain single alphabet (Schneider et al. 1990).

**Kullback-Leibler divergence (relative entropy)**

Above described uncertainty and information content assume uniform distribution of all alphabets. However, in realistic conditions, background nucleotide composition (e.g., genomic G+C content) varies among species, and distribution is even more diverse for protein sequences. In order to account for this heterogeneity of the innate distributions of the alphabets, relative entropy $H(P_i \| \pi)$ or Kullback-Leibler divergence from given background distribution $\pi$ is derived as follows:

$$H(P_i \| \pi) = H(P_i, \pi) - H(P_i) = \sum_{j \in M} P_{ij} (\log_2 P_{ij} - \log_2 \pi_j)$$

where $\pi_j$ is the background frequency of alphabet $j$. Note that $H(P_i \| \pi) = H(P_i)$ when $\pi = \frac{1}{|M|}$. Sequence Logo for protein motifs is developed using Kullback-Leibler divergence by Schuster-Böckler et al. (2004).

Although Kullback-Leibler divergence is commonly utilized to quantify the conservation in amino acid sequences, one should note that this is a divergence measure from the background distribution and it is not an information measure (Schneider 1999). This measure is called “divergence” since the calculation is asymmetric and therefore not sufficient as a distance measure. Moreover, Kullback-Leibler divergence can result in values exceeding the maximum uncertainty $\log_2 |M|$, and consequently it is inappropriate to be considered using the unit of “bits”.

**z-score cutoff**

Simplest means for the statistical testing of the significance of conservation is the use of z-score (also called the standard score), defined as:

$$z = \frac{P_i - \mu}{\sigma}$$

where $\mu$ is the mean and $\sigma$ is the standard deviation of all $P_{ij}$ for the most frequent alphabet $j$. The z-score represents how many standard deviations a raw score deviates from the mean. For example, representation of most frequent alphabet scoring $z > 2$ with upper case letters and those $2 > z > 1$ with lower case letters for regions surrounding the start codon in *Escherichia coli* K12 shown in Fig. 2 results in a following putative binding site: --a-ggGga---a--ATGaa-aa. Note that the z-score statistics can be used to identify significant positions within a binding site, but this is different from the amount of sequence conservation computed by the information theory. Comparison of indices based on information theory with z-score cut-off is discussed in a work using human cDNA sequences and G-language GAE (Aarakawa et al. 2005b).

**ANALYSIS OF NUCLEOTIDE COMPOSITION BIAS**

A genome is primarily shaped by the requirements of its coded genes, but at the same time, it is also highly organized as a functional medium that undergo replication, especially in fast growing bacteria where doubling time is in the order of less than a couple of hours. Circular bacterial chromosomes have single finite origin of replication from which replication forks progress bidirectionally, until the two forks meet at the replication origin and terminus, where the two replication origins correspond to the leading strand of one replication arm and the lagging strand of another arm, respectively (Lobry et al. 2003). Because of the discontinuous strand synthesis in the lagging strand, mutational bias in the two strands of DNA molecule result in the asymmetry in nucleotide composition (Lobry 1996b,Frank et al. 1999; Lobry et al. 2002). This characteristic genomic polarity can be visualized by plotting the relative abundance of C over G along the genomic positions, which is known as the GC skew graph (vide infra). FtsK translocase senses this genomic asymmetry by Schuster-Böckler et al. (2004).
computational asymmetry by recognizing KOPS oligomers in *E. coli*, in order to identify the dif sequence located near the replication terminus targeted by XerCD during chromosome dimer resolution (Peralis et al. 2000, 2001; Levy et al. 2005; Pease et al. 2005; Bigot et al. 2007). Accordingly, regions surrounding the replication terminus are enriched in skewed oligomers (Hendrickson et al. 2006), and coupled with the A+T enrichment around the replication terminus, background nucleotide composition bias affect the codon usage of genes depending on their locations within the genome (Daubin et al. 2003).

Near the replication origin, genes are preferentially located in the leading strand, especially for longer genes, operons, highly expressed genes, and essential genes (McLean et al. 1998; Rocha et al. 2003 Omont et al. 2004; Price et al. 2005). These tendencies are speculated to be formed by avoiding head-on collisions of DNA and RNA polymerases in the interplay of replication of transcription (Brewer 1988; Liu et al. 1995), and also due to the gene dosage effects of fast growing bacteria where multiple rounds of replication take place, especially for genes related to transcription and translation (Ardell et al. 2005; Couturier et al. 2006). In light of the many replication-related constraints that govern the chromosomal organization in bacterial genomes, it is essential to have the knowledge of accurate positions of replication origin and terminus, and subsequently the sequences for leading and lagging strands. Computational prediction of replication origin is a common practice in genome projects as a cost effective and sufficiently accurate alternative to experimental methods, and most bacterial genome projects use putative replication origin as the first base position when submitting the sequence data to public repositories.

**GC skew analysis**

GC skew is defined as the excess of C over G normalized by the G+C content \((C - G) / (C + G)\) in a given region (Lobry 1996). By graphing the GC skew values continuously along the genome sequence using sliding windows (for example, 10000 bp), most bacterial genomes are divided into C-rich lagging strand and G-rich leading strand (Fig. 3B). Replication origin and terminus are located in the vicinity of the shift points between two strands. Cumulative graph of GC skew is a frequently utilized alternative to clarify the shift points, where the maximum and minimum points correspond to the replication origin and terminus, respectively (Grigoriev 1998) (Fig. 3C). GC skew can be observed in both coding and intergenic regions, and sometimes GC skew of third codon positions is preferred to avoid the compositional bias of the coding regions (Frank et al. 2000). Similarly derived AT skew \((A - T) / (A + T)\) shows less significant polarity compared to GC skew, but the use of keto excess \((G + T - C - A) / (A + T + G + C)\) or purine excess \((G + A - C - T) / (A + T + G + C)\) is suggested to be more accurate in prediction of the replication origin and terminus for some bacterial species (Freeman et al. 1998). All of these measures are partial projections of the DNA walk diagrams, a pseudo-random walk representation of all nucleotides in a sequence, which is the trail drawn by moving a pixel in the direction of the type of nucleotide (in Fig. 3A, A-up, T-down, G-right, C-left).

**Prediction of replication origin and terminus**

Sequence-based prediction of replication origin and terminus identifies the peak positions of the aforementioned cumulative skew graphs. Oriloc is a popular implementation of this kind of algorithm, which detects the intersection of the DNA walk trajectory and its linear regression based on the nucleotide content of third codon positions (Frank et al. 2000). Although these methods are sufficiently accurate especially for replication origin, prediction of replication terminus usually has an error margin of around 10 kbp from the experimentally identified sites, due to high insertion and horizontal transfer rates (Moszer et al. 1991). To improve
the prediction accuracy for terminus regions and to clarify the skew shift points under the presence of background “noise”, low-pass filtering using Fast Fourier Transform (FFT), a common method to reduce innate noise in image or signal processing disciplines, has been proven to be successful (Arakawa et al. 2007a). In low-pass filtering, a given discrete signal with length \( N, f(n), n = 0, 1, \ldots, N - 1 \), at frequency \( k \), is transformed to frequency domain representation by FFT as follows,

\[
F(k) = \sum_{n=0}^{N-1} f(n)e^{-2\pi ikn/N},
\]

where \( i = \sqrt{-1} \). The power spectrum \( PS(k) \) of \( F(k) \) is subsequently obtained as follows,

\[
PS(k) = |F(k)|^2, \quad k = 0, 1, 2, \ldots, N - 1
\]

at each frequency \( k \). Since noise in data are distributed in the high frequency domain and replication-related selection should belong in low frequency domain considering the global nature of replication, zero-ing the power spectrum of the high frequency regions (thus “low-pass”) and reverse transforming the spectrum regenerates a skew graph with reduced noise. Obtaining the peak positions of the noise-filtered cumulative skew graph results in better prediction.

In addition to the skew-based methods, locations of cis-acting sequence elements related to replication are typically combined to support the computational prediction. For example, DnaA boxes \((5'-TTATTCACA-3' \) in \( E. coli \)) where DnaA proteins bind to unwind the DNA molecule in order to initiate the replication fork are indicative of the position of replication origin (Kaguni 1997), and the orientation of Ter sites where Tus proteins bind to block the replication fork progression only in one direction (Hill 1992) as well as the previously described \( disf \) sequence help identify the replication terminus. In bacterial plasmids, iteron sequences \((5'-TGAGGG G/A C/T-3' \) are indicative of replication origins (Haines et al. 2006). All methods described in this section include the identification of these sequence features are implemented in G-language GAE (See Table 2 for complete listing).

### GC Skew Index

Although GC skew is commonly observed in a wide variety of bacterial species, the “degree” or “visibility” of the skew is quite diverse. For example, slow growing bacteria such as Cyanobacteria and Mycoplasma exhibit only weak skews, and archaea do not show visible skew due to their different replication machinery (Daubin et al. 2003). Suitability of GC skew-based prediction methods for replication origin and terminus significantly relies on the clarity of GC skew, and therefore it is useful to quantify the degree of skew to allow comparative studies. GC Skew Index (GCSI) is a quantitative indicator for this purpose, calculated by combining the spectrum ratio \((SR) \) between 1Hz spectrum and the average of all spectra of 2Hz and above of FFT, and the Euclidean distance \((dist) \) between the two vertices of cumulative graph. Here \( SR \) captures the fitness of the “shape” of GC skew graphs, and \( dist \) measures the degree of bias (Arakawa et al. 2007b). GCSI is normalized to range from 0 to 1, and genomes with GCSI < 0.05 have no observable skew (majority of archaeal genomes belong to this category). \( E. coli \) has GCSI of around 0.1, above which clear shift points can be discerned. Therefore, GCSI should be a useful criterion to test the applicability of skew-based predictions, although one should also note that GCSI is not necessarily a measure of replication selection (Arakawa et al. 2007b).

### G+C Content

G+C content is the percent of guanine and cytosine in the nucleotide sequence, expressed as \( 100 \times (G + C)(A + T + G + C) \). Genomic G+C content varies widely among different bacterial species, and this variation is most pronounced at the third position of codons because the first two positions of codons are constrained by protein-coding requirements (Muto et al. 1987). For example, among 80 bacterial species tested by Sharp et al. (2005), genomic G+C content ranged from 22 to 72%, whereas G+C content at synonymously variable third positions ranged from 9 to 93%. Various factors have been proposed as determinants of G+C content, including genome-wide mutational bias toward G+C or A+T (Sueoka 1962), higher energy cost and limited availability of G+C over A+T (Rocha et al. 2002), increment in G+C in aerobiosis (Naya et al. 2002), and

### Table 2 Programs for genomic compositional analysis and sequence pattern searches implemented in G-language GAE.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>gcskew</td>
<td>Calculates and graphs the GC skew of the genome. By specifying the optional</td>
</tr>
<tr>
<td></td>
<td>parameter, this method can also graph AT skew, purine and keto excess, and</td>
</tr>
<tr>
<td></td>
<td>cumulative skew.</td>
</tr>
<tr>
<td>genomicskew</td>
<td>Graphs the GC skew of whole genome, coding regions, GC3, and intergenic</td>
</tr>
<tr>
<td></td>
<td>regions. Optionally shows AT skew or purine / keto excess for those regions.</td>
</tr>
<tr>
<td>dna走</td>
<td>Graphs the DNA walk of given sequence.</td>
</tr>
<tr>
<td>find_ori</td>
<td>Predicts the replication origin and terminus by identifying the peaks of</td>
</tr>
<tr>
<td></td>
<td>cumulative GC skew graph at single base-pair resolution.</td>
</tr>
<tr>
<td>rep_ori</td>
<td>Optionally uses AT skew or purine / keto excess for prediction, and the use</td>
</tr>
<tr>
<td></td>
<td>of FFT-based noise reduction filtering can be specified.</td>
</tr>
<tr>
<td>gcsi</td>
<td>Quantifies the degree of skew (GCSI).</td>
</tr>
<tr>
<td>leading_strand</td>
<td>Returns the sequence of leading strand.</td>
</tr>
<tr>
<td>query_strand</td>
<td>Given a position and direction of the strand, returns whether it is on the</td>
</tr>
<tr>
<td></td>
<td>leading or lagging strand.</td>
</tr>
<tr>
<td>set_gc3</td>
<td>Sets the GC3 information for all genes.</td>
</tr>
<tr>
<td>genes_from_ori</td>
<td>Retrieves gene names in the order of distance from the origin.</td>
</tr>
<tr>
<td>dist_in_ec</td>
<td>Calculates the distance of given position from the origin.</td>
</tr>
<tr>
<td>gcwin</td>
<td>Calculates and graphs the GC content along the chromosome.</td>
</tr>
<tr>
<td></td>
<td>Optionally calculates AT content.</td>
</tr>
<tr>
<td>signature</td>
<td>Calculates the oligonucleotide relative abundance (genomic signature).</td>
</tr>
<tr>
<td>palindrome</td>
<td>Searches for palindrome sequences of given length.</td>
</tr>
<tr>
<td>find_diff</td>
<td>Searches for disf sequence.</td>
</tr>
<tr>
<td>find_ter</td>
<td>Searches for Ter sites</td>
</tr>
<tr>
<td>find_dnaAbox</td>
<td>Searches for dnaA-boxes</td>
</tr>
<tr>
<td>find_iron</td>
<td>Searches for Iterons</td>
</tr>
<tr>
<td>oligomer_search</td>
<td>Searches for given oligomer. Degenerate nucleotide code or regular</td>
</tr>
<tr>
<td></td>
<td>expressions can be used.</td>
</tr>
</tbody>
</table>

- **G+T content**
- **G+C content**
horizontal DNA transfer among distantly related species with different genomic G+C contents (Lawrence et al. 1997). To identify putative foreign genes, G+C content is determined for overall protein-coding regions and/or at different positions of codons (Lawrence et al. 1997 Garcia-Vallve et al. 2000). To identify genomic islands (clusters of foreign genes), G+C content is computed using sliding windows (Karlin 2001), and this is sometimes also useful to identify the putative coding regions especially in A+T rich genomes (Chen et al. 2004b).

**Genomic signature**

Karlin and his coworkers proposed that each organism has its characteristic “genomic signature” defined as the ratios between the observed and expected frequencies of dinucleotides (dinucleotide relative abundances) (Karlin et al. 1995; Karlin et al. 1998a). The dinucleotide relative abundance value ($\rho_{XY}$) is calculated as:

$$\rho_{XY} = \frac{f_{XY}}{f_{X}f_{Y}}$$

where $f_{X}$ and $f_{Y}$ denote the frequency of the mononucleotide $X$ and $Y$ respectively, and $f_{XY}$ denotes the frequency of the dinucleotide $XY$, computed from the sequence concatenated with its inverted complement sequence. Because the genomic signature is relatively constant throughout the genome and similar between closely related species, it has been used to construct phylogenetic trees (Coenye et al. 2003; Coenye et al. 2004; van Passel et al. 2006) and to detect anomalous genomic regions such as genomic islands (Karlin 2001; van Passel et al. 2005). The analysis of different word length (e.g., 4-letter words = tetranucleotide) in different size of sliding window (Dufraine et al. 2005) can be implemented in G-language GAE.

**ANA LYSIS OF SYNONYMOUS CODON USAGE BIAS**

Much genetic code is degenerate, meaning that most amino acids are encoded by more than one codon (triplet of nucleotides); these synonymous codons usually differ by one nucleotide in the third position. Synonymous codons are not used with equal frequency, and their usage varies among different species and also among genes within the same genome (Sharp et al. 1988). Different factors have been proposed to explain variations in synonymous codon usage among genes, including genome-wide mutational bias (shaping intergenomic variation in G+C content) (Chen et al. 2004a), natural selection linked to optimal growth temperature (Lynn et al. 2002; Lobry et al. 2006), horizontal gene transfer among distantly related species (shaping intragenomic variation in G+C content) (Lawrence et al. 1997; Garcia-Vallve et al. 2000), strand-specific mutational bias (shape the dinucleotide $XY$), computed from the sequence concatenated with its inverted complement sequence. Because the genomic signature is relatively constant throughout the genome and similar between closely related species, it has been used to construct phylogenetic trees (Coenye et al. 2003; Coenye et al. 2004; van Passel et al. 2006) and to detect anomalous genomic regions such as genomic islands (Karlin 2001; van Passel et al. 2005). The analysis of different word length (e.g., 4-letter words = tetranucleotide) in different size of sliding window (Dufraine et al. 2005) can be implemented in G-language GAE.

**Table 3 Programs for codon analysis implemented in G-language GAE.**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>codon_usage</td>
<td>Displays the codon table of the given genome or specified gene.</td>
</tr>
<tr>
<td>codon_mva</td>
<td>Performs multivariate analyses of codon usage data, and analyzes correlations between the axes and other gene features such as G+C content and GC skew.</td>
</tr>
<tr>
<td>enc</td>
<td>Calculates the effective number of codons (Nc).</td>
</tr>
<tr>
<td>cbi</td>
<td>Calculates the codon bias index (CBI).</td>
</tr>
<tr>
<td>icdi</td>
<td>Calculates the intrinsic codon deviation index (ICDI).</td>
</tr>
<tr>
<td>Ew</td>
<td>Calculates the weighted sum of relative entropy (Ew).</td>
</tr>
<tr>
<td>P2</td>
<td>Calculates the P2 index.</td>
</tr>
<tr>
<td>fop</td>
<td>Calculates the frequency of optimal codons (Fop).</td>
</tr>
<tr>
<td>w_value</td>
<td>Calculates the relative adaptiveness (W) of each codon.</td>
</tr>
<tr>
<td>cai</td>
<td>Calculates codon adaptation index (CAI) for each gene.</td>
</tr>
<tr>
<td>phx</td>
<td>Calculates the expression measure, $E_{phx}$, to identify predicted highly expressed (PHX) genes.</td>
</tr>
</tbody>
</table>

Multivariate analyses of codon usage data

Multivariate analyses methods, such as correspondence analysis (Grantham et al. 1980) and principal component analysis (PCA) (Canaya et al. 1996), are often used to identify gene features contributing to the variations in synonymous codon usage among genes. Different kinds of codon usage data have been applied to these multivariate analysis methods (Perriere et al. 2002; Suzuki et al. 2005). Of the five codon usage data ($R_0-R_4$), only $R_4$ is independent of all three biases: (i) gene length, (ii) amino acid composition, and (iii) codon degeneracy. Indeed, PCA of $R_4$ data (PCA-$R_4$) is not affected by any of these biases (Suzuki et al. 2005). Consequently, PCA-$R_4$ is more effective than the other four methods at detecting gene features related to synonymous codon usage variations such as $G$+$C$ content ($G$+$C$)/($A$+$T$ + $G$+$C$) at the third codon position (GC3) and GC skew ($C$ - $G$)/(C - $G$) at the third codon position (Fig. 4).

**Representation of codon usage data**

Five different kinds of representations of codon usage data (termed here $R_0$-$R_4$) have been used in codon usage studies (Perriere et al. 2002; Suzuki et al. 2005). For a single gene or a group of genes, the value of the jth amino acid ($x_j$) is defined as:

For $R_0$, $x_j = n_j$

For $R_1$, $x_j = \frac{n_j}{\sum_{i=1}^{n} n_i}$

For $R_2$, $x_j = \frac{n_j}{\sum_{i=1}^{n} n_j}$

For $R_3$, $x_j = \frac{1}{k_j} \sum_{i=1}^{n} n_i$

For $R_4$, $x_j = \frac{n_j}{n_{max}}$

where $n_j$ is the number of jth codon for the ith amino acid, $k_j$ is the degree of codon degeneracy for the ith amino acid (e.g., $k_j = 2$ for cysteine and $k_j = 6$ for arginine), and $n_{max}$ is the number of the most frequently used synonymous codon for the ith amino acid. The codon usage data $R_0$, $R_2$, $R_3$, and $R_4$ are also called as the absolute codon frequency (AF), the relative codon frequency (RF), the relative synonymous codon usage (RSCU), and the relative adaptiveness (W) of each codon, respectively (Sharp et al. 1986, 1987; Perriere et al. 2002).
Fig. 5 shows the plots of first and second axis scores obtained by PCA-R$_4$ for all individual genes in two fast growing bacteria *E. coli* and *Bacillus subtilis* as examples. The distribution of points reveals two horns: that corresponding to constitutively highly expressed genes (encoding ribosomal proteins and elongation factors) and that corresponding to putative foreign genes (Medigue et al. 1991; Moszer et al. 1999).

**Measure of synonymous codon usage evenness**

Various measures of synonymous codon usage evenness have been proposed, including the ‘effective number of codons’ ($N_e$) (Wright 1990), the codon bias index (CBI) (Morton 1993), the intrinsic codon deviation index (ICDI) (Freire-Picos et al. 1994), Shannon uncertainty (entropy) from information theory ($H_s$) (Zeeberg 2002) and its modification, called the ‘weighted sum of relative entropy’ ($E_w$) (Suzuki et al. 2004). The entropy of the $i^{th}$ amino acid is defined as:

$$H_i = -\sum_{j=1}^{20} x_{ij} \log_2 x_{ij}$$

where $x_{ij}$ is the relative codon frequency ($R_i$) of the $j^{th}$ codon for the $i^{th}$ amino acid, and $k_i$ is the degree of codon degeneracy for the $i^{th}$ amino acid. $H_s$ and $E_w$ is calculated by combining the entropies from different amino acids:

$$H_s = \sum_{i=1}^{20} H_i$$

$$E_w = \sum_{i=1}^{20} \left( \frac{H_i}{\log_2 k_i} \right) p_i$$

where $p_i$ is the relative frequency of the $i^{th}$ amino acid in the protein. $E_w$ ranges from 0 (maximum bias) to 1 (maximum evenness). Because $E_w$ takes into account all three aspects of amino acid usage, i.e., (i) the number of different amino acids, (ii) their relative frequency, and (iii) their codon degeneracy, it is little affected by amino acid usage biases (Suzuki et al. 2004).

**Predicting gene expression level from codon usage**

Various methods of predicting gene expression level from codon usage bias have been proposed, including the P2 index (Gouy et al. 1982), the frequency of optimal codons (FOP) (Ikemura 1985), the codon adaptation index (CAI) (Sharp et al. 1987), and the expression measure, $E(g)$, for identifying predicted highly expressed (PHX) genes (Karlin et al. 2000). In some species, putative highly expressed genes (e.g., those encoding ribosomal proteins) do not have unusual codon usage, and thus codon usage cannot be used to predict gene expression levels (Grocock et al. 2002; Carbone et al. 2003). Therefore, to estimate the level of gene expression from codon usage bias, it is necessary first to check whether a genome shows evidence of translationally
selected codon usage bias by comparing codon usage of highly expressed genes with that of all genes (Henry et al. 2007), as shown in Fig. 5.

P2 index. The P2 index represents the proportion of codons conforming to the intermediate strength of codon-anticodon interaction energy rule of Grosjean and Fiers (Grosjean et al. 1982), and calculated as P2 = (WWC + SSU) / (WWY + SSY), where W = A or U, S = G or C, and Y = C or U (Gouy et al. 1982). It indicates the efficiency of the codon–anticodon interaction and has been used as an indicator of the presence of translational selection (von Samson-Himmelstjerna et al. 2003). In fast growing bacteria such as E. coli, highly expressed genes have high P2 values (0.7-0.9), while other genes have values close to 0.5 (Shields et al. 1987).

Frequency of optimal codons (FOP). In fast growing bacteria such as E. coli, highly expressed genes preferentially use optimal codons, which optimize the efficiency of translation (determined by tRNA availability and the efficiency of codon-anticodon pairing) (Ikemura 1985). FOP is defined as the number of optimal codons divided by the sum of the number of optimal and nonoptimal codons. FOP takes values from 0.0 (where no optimal codons are used) to 1.0 (where only optimal codons are used).

Codon adaptation index (CAI). CAI is a measure of the relative adaptiveness of the codon usage of a gene towards the codon usage of highly expressed genes (Sharp et al. 1987), and computed as:

\[
CAI = \exp \left( \sum_{i=1}^{20} \sum_{j=1}^{3} n_{ij} \ln x_{ij} \right)
\]

where \( n_{ij} \) is the number of the \( j \)th codon for the \( i \)th amino acid, \( k_i \) is the degree of codon degeneracy for the \( i \)th amino acid, and \( x_{ij} \) is the relative adaptiveness (\( R_{ij} \)) value of the \( j \)th codon for the \( i \)th amino acid in a reference set of highly expressed genes. Thus, CAI is defined as the geometric mean of the \( R_{ij} \) values, and ranges from 0.0 to 1.0. CAI can be used as a ‘universal’ measure of codon usage bias; that is, CAI values can be correlated with gene features other than gene expression level (e.g., GC content, GC skew, and so on) using a reference set of genes which is representative of the bias (Carbone et al. 2003, 2005).

Expression measure, \( E(g) \). Let \( x_{ij}(g) \) indicate the relative codon frequency (\( R_{ij} \)) value of the \( j \)th codon for the \( i \)th amino acid in the gene \( g \). The codon usage difference of the gene \( g \) relative to the gene group \( G \) is calculated by the formula (Karlin et al. 1998a, 1998b):

\[
B(g) = \sum_{i=1}^{20} p_i(g) \sum_{j=1}^{3} [x_{ij}(g) - x_{ij}(G)]
\]

where \( p_i(g) \) is the relative frequency of \( i \)th amino acid of the gene \( g \). Denoted by \( C \) is the collection of all protein genes, and by \( H \) the putative highly expressed genes (those encoding ribosomal proteins, translation/transcription processing factors, and chaperone/degradation proteins). The general form of the expression measure is:

\[
E(g) = \frac{B(g|C)}{B(g|H)}
\]

A gene \( g \) is deemed ‘predicted highly expressed’ (PHX) if \( B(g|H) \) is lower than \( B(g|C) \); i.e., \( E(g) \) exceeds 1.0 (Karlin et al. 2000, 2001b, 2003, 2005). A gene \( g \) is deemed ‘putative alien’ (PA) provided both \( B(g|H) \) and \( B(g|C) \) exceed the median value for all genes (Mrazek et al. 1999; Karlin et al. 2001a; Mrazek et al. 2001).

EXAMPLE WORKFLOW

G-language GAЕ is utilized in many fields of bioinformatics and computational biology, including genomics, software development for bioinformatics, systems biology, non-coding RNAs, and cis-acting sequence elements (see http://www.g-language.org/wiki/publications for a list of scientific publications citing G-language GAЕ). Nonetheless, as described thus far, G-language GAЕ is especially comprehensive for the analysis of sequence conservation by information theory, genomic compositional asymmetry, and synonymous codon usage bias, in comparison with similar software packages in each of these areas. Although programming is inevitable in bioinformatics, initial screening processes for these areas may be achieved only with simple combination of implemented programs within the interactive shell environment. G-language GAЕ supports the creation of workflows using implemented methods with minimal scripting, and several published works provide such workflow files (GCF format) as supplementary materials so that the procedure can be reused and configured (Sato et al. 2003; Arakawa et al. 2005b; Yachie et al. 2006).

Here we describe one example workflow that we have actually used in the screening process of a recent work that analyzed the correlation of gene positioning relative to the replication origin and the gene features (GC3, gene length, predicted gene expression level, general codon usage bias, essentiality, and functional classification) within circular bacterial chromosomes (Arakawa et al. 2007b). Workflow diagram for the initial screening to see if these gene features are correlated with their relative positions from replication origin is depicted in Fig. 6. Firstly, a GenBank format file is automatically downloaded, parsed, and loaded upon calling “load” function with appropriate RefSeq accession number of E. coli. Thus obtained genome data object is then annotated for the gene features of interest: gene essentiality using Profiling of E. coli Chromosome database (Hashimoto et al. 2005), GC3 of each gene, functional classification using the NCBI COG database through “set_gpac” program (Tatusov et al. 2001), CAI calculated using ribosomal proteins as reference (predicted gene expression level), and CAI calculated using all proteins as reference (general codon usage bias). Genes are then ordered by their relative distances from replication origin, after the prediction of replication origin and terminus using GC skew-based methods and database searches. Finally, gene features and the relative positions from the origin are statistically compared and graphed to ease the interpretation of results. G-language GAЕ is equipped with many basic statistics tools, so the distribution of values is first tested for normality using Kolmogorov-Smirnov Lilliefors test, and then based on this result, the degree of correlation between two variables is quantified using Spearman’s rank correlation coefficient. Although further analysis in detail requires programming, all tasks in this workflow can be achieved with implemented methods only, within the interactive shell.

CONCLUSIONS AND OUTLOOK

G-language GAЕ is unique among many bioinformatics workbenches, since it provides numerous genome analysis tools and algorithms in the form of programming interface, examples of those especially related to bacterial genomes analysis are described in this review. Programming is the central means in computational biology research, and availability of useful algorithms as APIs allows maximum flexibility and freedom for computational molecular biologist in combination with basic interface libraries provided by Bio* projects. As exemplified by the methods for the analysis of binding sites, codons, and nucleotide composi-
tions, G-language GAE is implemented with several algorithms for a given biological problem, therefore the users can choose the most suitable method for their needs and objectives. Most analysis programs in G-language GAE output graphical results in the form of graphs and diagrams in order to ease the interpretation by researchers, and they are also equipped with optional parameters with which analyses can be fine-tuned and configured.

We would like to stress the fact that any daily research work including computational genome analysis is mostly comprised of the trial-and-error processes, where researchers explore numerous datasets, tools and algorithms, and their parameters in combination, in order to best solve their biological problems. Therefore, to make the research process more efficient, a workbench for bioinformatics should support the heuristic nature of research routines. G-language shell interface with persistent memory, help command, logging as Perl script, tab-completion of file and program names, and interactivity coupled with the visual output and optional parameters is entirely designed for this
Purpose. Workflows for bioinformatics, the result of such trial-and-errors, are essential to allow reuse and sharing of methods as is commonly done in “wet” biology with experimental protocols, but the major bottleneck in current computational biology, in our opinions, is in the heuristic process in the construction of workflows.

In addition to enrolling more methods and documentations within the software system, we here propose three arguments for why bioinformatics software environment can possibly improve to make the heuristic processes more efficient: web-service, scientific visualization, and user interface. Rapid accumulation of sequence data in public databases exceeds the rate of the Moore’s law of transistors (Benson et al. 2007), and this is expected to further accelerate with the introduction of next-generation sequencers (Blow 2007). Considering the amount of data required to transfer over internet and to mirror locally besides the regular updates on annotations, software installation and compatibility, and wealth of computational resource typically equipped in large database providers, it is often useful to take advantage of web-services in heuristic screening (Fox et al. 2007). Standardization of input/output data types and method classification, redundancy in service provision, distribution of computation utilizing grid environment, and client software to support these frameworks would be necessary to make full use of the hundreds of services in concert (Stein 2002). Scientific visualization is gaining much attention in light of the complex nature of omics data and it has been successful to display results of biological research (Ball 2002; Arakawa et al. 2005a; Kono et al. 2006), but visualization that aids the heuristic processes of scientific research is less explored. Visualization that does not necessarily show the final results but that allows researchers to identify certain meanings and patterns within the huge masses of information, possibly through frequent interactions by researchers according to their heuristics, would contribute to both of computational and experimental biology. In terms of software engineering, these new frontiers should be coupled with better user interfaces that give prompt response (which may require greater computational efficiency) upon user interaction to accelerate the heuristics of researchers. As a workbench for bioinformatics, these challenges are the current goals for G-Language Project.

Acknowledgements

This research is supported in part by the Grant-in-Aid for Young Scientists (Start-up, B), No. 19810021, 2007, and No. 20710158, 2008, from the Japan Society for the Promotion of Science (JSPS), and by the funds from Yamagata Prefectural Government and Tsuruoka City.

References

Arakawa K, Tomita M (2007b) Selection effects on the positioning of genes and gene structures from the interplay of replication and transcription in bacterial genomes. Evolutionary Bioinformatics 3, 279-286
Chen YT, Chang HY, Lai YC, Pan CC, Tsai SF, Peng HL (2004b) Sequencing and analysis of the large virulence plasmid pVBP of Klebsiella pneumoniae CIP143. Gene 337, 189-198
Coenye T, Vandamme P (2003) Extracting phylogenetic information from whole-genome sequencing projects: The lactic acid bacteria as a test case. Microbiology 149, 3507-3517


Wright F (1990) The 'effective number of codons' used in a gene. Gene 87, 23-29
