CodonGenie: optimised ambiguous codon design tools (#16109)

First revision

Please read the **Important notes** below, the **Review guidance** on page 2 and our **Standout reviewing tips** on page 3. When ready **submit online**. The manuscript starts on page 4.

Important notes

Editor

James Procter

Files	1 Tracked changes manuscript(s) 1 Rebuttal letter(s) 1 Figure file(s) 1 Table file(s) Please visit the overview page to **download and review** the files not included in this review PDF.
Declarations	No notable declarations are present

Review guidelines

Please read in full before you begin

How to review

When ready **submit your review online**. The review form is divided into 5 sections. Please consider these when composing your review:

1. **BASIC REPORTING**
2. **EXPERIMENTAL DESIGN**
3. **VALIDITY OF THE FINDINGS**
4. General comments
5. Confidential notes to the editor

You can also annotate this PDF and upload it as part of your review

To finish, enter your editorial recommendation (accept, revise or reject) and submit.

BASIC REPORTING

- ✓ Clear, unambiguous, professional English language used throughout.
- ✓ Intro & background to show context. Literature well referenced & relevant.
- ✓ Structure conforms to **PeerJ standards**, discipline norm, or improved for clarity.
- ✓ Figures are relevant, high quality, well labelled & described.
- ✓ Raw data supplied (see **PeerJ policy**).

EXPERIMENTAL DESIGN

- ✓ Original primary research within **Scope of the journal**.
- ✓ Research question well defined, relevant & meaningful. It is stated how the research fills an identified knowledge gap.
- ✓ Rigorous investigation performed to a high technical & ethical standard.
- ✓ Methods described with sufficient detail & information to replicate.

VALIDITY OF THE FINDINGS

- i Impact and novelty not assessed. Negative/inconclusive results accepted. *Meaningful* replication encouraged where rationale & benefit to literature is clearly stated.
- ✓ Data is robust, statistically sound, & controlled.
- ✓ Conclusions are well stated, linked to original research question & limited to supporting results.
- ✓ Speculation is welcome, but should be identified as such.

The above is the editorial criteria summary. To view in full visit **https://peerj.com/about/editorial-criteria/**

7 Standout reviewing tips

The best reviewers use these techniques

Tip	***Example***
Support criticisms with evidence from the text or from other sources	*Smith et al (J of Methodology, 2005, V3, pp 123) have shown that the analysis you use in Lines 241-250 is not the most appropriate for this situation. Please explain why you used this method.*
Give specific suggestions on how to improve the manuscript	*Your introduction needs more detail. I suggest that you improve the description at lines 57- 86 to provide more justification for your study (specifically, you should expand upon the knowledge gap being filled).*
Comment on language and grammar issues	*The English language should be improved to ensure that your international audience can clearly understand your text. I suggest that you have a native English speaking colleague review your manuscript. Some examples where the language could be improved include lines 23, 77, 121, 128 – the current phrasing makes comprehension difficult.*
Organize by importance of the issues, and number your points	*1. Your most important issue* *2. The next most important item* *3. …* *4. The least important points*
Give specific suggestions on how to improve the manuscript	*Line 56: Note that experimental data on sprawling animals needs to be updated. Line 66: Please consider exchanging "modern" with "cursorial".*
Please provide constructive criticism, and avoid personal opinions	*I thank you for providing the raw data, however your supplemental files need more descriptive metadata identifiers to be useful to future readers. Although your results are compelling, the data analysis should be improved in the following ways: AA, BB, CC*
Comment on strengths (as well as weaknesses) of the manuscript	*I commend the authors for their extensive data set, compiled over many years of detailed fieldwork. In addition, the manuscript is clearly written in professional, unambiguous language. If there is a weakness, it is in the statistical analysis (as I have noted above) which should be improved upon before Acceptance.*

CodonGenie: optimised ambiguous codon design tools

Neil Swainston Corresp., [1] , **Andrew Currin** [1] , **Lucy Green** [1] , **Rainer Breitling** [1, 2] , **Philip J Day** [3] , **Douglas B Kell** [1, 2]

[1] Manchester Centre for Synthetic Biology of Fine and Speciality Chemicals (SYNBIOCHEM), University of Manchester, Manchester, United Kingdom
[2] School of Chemistry, University of Manchester, Manchester, United Kingdom
[3] Faculty of Biology, Medicine and Health, University of Manchester, Manchester, United Kingdom

Corresponding Author: Neil Swainston
Email address: neil.swainston@manchester.ac.uk

CodonGenie, freely available from http://codon.synbiochem.co.uk , is a simple web application for designing ambiguous codons to support protein mutagenesis applications. Ambiguous codons are derived from specific heterogeneous nucleotide mixtures, which create sequence degeneracy when synthesised in a DNA library. In directed evolution studies, such codons are carefully selected to encode multiple amino acids. For example, the codon NTN, where the code N denotes a mixture of all four nucleotides, will encode a mixture of phenylalanine, leucine, isoleucine, methionine and valine. Given a user-defined target collection of amino acids matched to an intended host organism, CodonGenie designs and analyses all ambiguous codons that encode the required amino acids. The codons are ranked according to their efficiency in encoding the required amino acids while minimising the inclusion of additional amino acids and stop codons. Organism-specific codon usage is also considered.

CodonGenie: optimised ambiguous codon design tools

[1,*]Neil Swainston, [1]Andrew Currin, [1]Lucy Green, [1,2]Rainer Breitling, [3]Philip J Day, [1,2]Douglas B Kell

[1]Manchester Centre for Synthetic Biology of Fine and Speciality Chemicals (SYNBIOCHEM), Manchester Institute of Biotechnology, University of Manchester, Manchester M1 7DN, United Kingdom.

[2]School of Chemistry, University of Manchester, Manchester M13 9PL, United Kingdom.

[3]Faculty of Biology, Medicine and Health, University of Manchester, Manchester M13 9PL, United Kingdom.

[*]Corresponding author.

Abstract

CodonGenie, freely available from http://codon.synbiochem.co.uk, is a simple web application for designing ambiguous codons to support protein mutagenesis applications. Ambiguous codons are derived from specific heterogeneous nucleotide mixtures, which create sequence degeneracy when synthesised in a DNA library. In directed evolution studies, such codons are carefully selected to encode multiple amino acids. For example, the codon NTN, where the code N denotes a mixture of all four nucleotides, will encode a mixture of phenylalanine, leucine, isoleucine, methionine and valine. Given a user-defined target collection of amino acids matched to an intended host organism, CodonGenie designs and analyses all ambiguous codons that encode the required amino acids. The codons are ranked according to their efficiency in encoding the required amino acids while minimising the inclusion of additional amino acids and stop codons. Organism-specific codon usage is also considered.

Introduction

Site-directed mutagenesis of DNA is an established technique of generating libraries of DNA variants in a controlled manner, and has applications in a range of fields, primarily that of protein engineering (Jäckel, Kast & Hilvert, 2008), but also in more fundamental research including the study of sequence-to-fitness relationships (Hietpas et al., 2011). The design of mutant protein libraries typically involves a manual process in which required sites for mutation are selected and ambiguous codons (those containing mixtures of nucleotides) designed to introduce controlled variation in these positions.

In this process, one may wish to design a codon to specify any subset of amino acids in a given position. Since each amino acid may be included in the subset or otherwise, the number of possible subsets is $2^{20} - 1$, i.e. there are 1,048,575 possible subsets of 20 amino acids. (Each of the sets can be represented by a 20-digit binary number, where a one at position *n* indicates that amino acid *n* is included in the set, and a zero indicates that it is absent. There are 2^{20} such numbers, but one of them represents the empty set and is thus not counted here.) Not all of these 1,048,575 subsets of 20 amino acids are uniquely designable using ambiguous codons, of which there are only 3375. (There are 15 (=2^4 – 1) relevant nucleotide codes ("letters"), ranging from the completely unambiguous `A`, `C`, `G` and `T` representing a single nucleotide, to the completely ambiguous `N` representing all 4 nucleotides (Cornish-Bowden, 1985). There are 15^3 = 3375 triplet codons that can be assembled from this 15-letter alphabet of ambiguous codes, compared to the 4^3 = 64 codons that can be constructed from the standard 4-letter alphabet of unambiguous nucleotides.)

Given the degeneracy of the codon table, there are often multiple ways to encode a chosen set of amino acids. The experimenter must a) decide if it is feasible to encode all desired amino acids (Mena & Daugherty, 2005); b) determine whether this creates an acceptable number of sequence combinations (depending on screening capability and throughput) (Currin et al., 2015; Kille et al., 2013; Lutz, 2010; Pines et al., 2015); and c) consider the codon usage of the organism to be used (Nakamura, Gojobori & Ikemura, 2000). It ~~therefore~~ follows that the design of ambiguous codons is non-trivial.

CodonGenie is therefore introduced to provide a quick and easy-to-use means of designing optimal ambiguous codons, considering the above parameters according to the user input, and ranking the ambiguous codons with respect to their suitability for expression in a target host organism. The tool is designed to be both human- and computer-readable, providing both a simple web browser interface and a RESTful webservice API.

Materials & Methods

Algorithm

The standard codon table is such that 17 of the 20 naturally occurring amino acids are encoded by codons with fixed bases in the first and second positions, with the third "wobble"-position allowing variation that accounts for the degeneracy of the DNA code. Determining optimal ambiguous codons for combinations of amino acids involves the following process, which is optimized for computational efficiency, compared to a brute-force examination of all possible ambiguous codons:

Align the first two positions and select the most specific ambiguous bases to encode the alignment. For example, with the combination asparagine and isoleucine (encoded by `AA[CT]` and `AT[ACT]` respectively), the alignment of the first two positions is `A[AT]`, i.e. `AW`.

All combinations of aligned wobble positions are calculated, i.e. `[CA]`, `[CC]`, `[CT]`, `[TA]`, `[TC]`, `[TT]`. These are then collapsed into unique sets, in this example giving `[CA]`, `C`, `[CT]`, `[TA]` and `T`.

The first two and wobble position bases are combined to produce candidate ambiguous codons, which are scored as described below.

Three amino acids (leucine, arginine and serine) cannot be simply encoded by codons with fixed bases in the first and second positions. (For example, both `CTN` and `TT[AG]` encode leucine.) For combinations including these more complex residues, the above algorithm is performed for each encoding and the results combined.

Note that CodonGenie returns not only the most "specific" ambiguous codons, that is, the codons that provide the fewest DNA variants whilst encoding all target amino acids. Providing results that include less specific ambiguous codons, which may also encode additional amino acids, allows the user to ~~perform a~~ trade~~-off~~ ~~between~~ library size ~~and~~ codon specificity, depending on the experimental objective. A smaller library is generally advantageous for screening purposes, but may contain codons that are unfavoured by the target host organism.

Scoring

The goal of the scoring scheme is to preferentially rank the most efficient ambiguous codons. That is, the ambiguous codons that encodes all of the required amino acids while minimising the encoding on non-desired amino acids.

The score for an ambiguous codon is therefore defined as the mean of the *value*, v_i, of each of the codons that it encodes. For codons that encode required amino acids, v_i is the ratio of the frequency of the codon f_i and the frequency of the most frequent synonymous codon f_j for the amino acid that it encodes. For codons that encode non-required amino acids, v_i is zero.

$$\text{score} = \frac{1}{|C|}\sum_{i \in C} v_i$$, where

$$v_i = \begin{cases} \frac{f_i}{\max(\{f_j : j \in S_i\})} & i \in R \\ 0 & i \notin R \end{cases}$$

$C = \{\text{all variants of ambiguous codon } c\}$

$A = \{\text{target amino acids}\}$

a_i : amino acid encoded by codon $i \in C$

f_i : codon usage frequency of codon $i \in C$

$S_i = \{j : a_j = a_i\}$ Set of synonymous codons of codon i

$R = \{i \in C : a_i \in A\}$ Set of codon variants of c encoding target amino acids

This scoring algorithm thus achieves a principled trade-off between codon specificity, library size and codon favourability (according to the codon usage preferences of the target organism).

Web service access

CodonGenie also offers a RESTful web service interface, supporting its integration with software pipelines. The Design method can be accessed by specifying required amino acids and required host organism (as an NCBI Taxonomy id (Federhen, 2012)) as follows:

http://codon.synbiochem.co.uk/codons?aminoAcids=DE&organism=4932

Similarly, the Analyse method can be accessed by specifying a variant codon and the required organism:

http://codon.synbiochem.co.uk/codons?codon=NSS&organism=4932

CodonGenie also provides web service interfaces for accessing supported organisms. The first allows all organisms to be listed, showing NCBI Taxonomy id and name, and the second allows the collection to be searched according to a given term:

http://codon.synbiochem.co.uk/organisms/
http://codon.synbiochem.co.uk/organisms/escher

In all cases, results are returned in json format.

Distribution

The web application is freely available from http://codon.synbiochem.co.uk. CodonGenie is written in Python (using the Flask framework) and HTML / Javascript (using the Bootstrap and AngularJS libraries) and is packaged as a Docker application for ease of deployment. Source code is available from https://github.com/synbiochem/CodonGenie.

Results and Discussion

CodonGenie provides a simple web interface affording two functions: a) the design, and b) the analysis of ambiguous codons. Considering the Design module, the user specifies the combination of amino acids to be encoded and an organism in which the library will be expressed. The codon usage table is automatically extracted from the Codon Usage Database (Nakamura, Gojobori & Ikemura, 2000), which as of May 2017 provided support for 35,792 organisms. CodonGenie then calculates suitable ambiguous codons and presents these in an interactive table (see Figure 1).

The Analyse module provides the functionality of checking an existing ambiguous codon. Users specify a variant codon and required host organism, and the results returned indicate which amino acids are encoded along with their codon usage frequency.

The benefit of CodonGenie can be exemplified by the design of an ambiguous codon to encode non-polar amino acids phenylalanine, leucine, isoleucine, methionine and valine. A simple and widely used ambiguous codon to encode this subset is `NTN`, which equates to 16 DNA variants. However, CodonGenie identifies that these same amino acids can be encoded by the `DTK` codon (where `D` denotes `[AGT]` and `K` denotes `[GT]`) using 6 variants. Selecting `DTK` therefore means fewer enzyme variants need to be screened to test all sequence combinations. This benefit is particularly significant when encoding multiple variant codons. For example, when using 3 `DTK` codons the library size is reduced from 4096 (16^3) to 213 (6^3) combinations.

An example of the importance of considering codon usage of the target host organism can be seen when considering the design of an ambiguous codon to encode the set of five non-polar amino acids (`F`, `I`, `L`, `M` and `V`) considered above. For *E. coli*, the preferred codon is `DTK` (`ATG|T|GT`), with a score of 0.88. `DTS` (`ATG|T|GC`) also encodes all five amino acids using 6 variants, but with a score of 0.68. In *Streptomyces coelicolor* – a commonly used host for antibiotic production (Pickens et al., 2011), the ranking is reversed, with `DTS` being preferred with a score of 0.79, substantially higher than that of 0.29 for `DTK`. The reason for this can be found in the codon usage frequencies of each of these organisms, as shown in Table 1: The codons `DTK` and `DTS` differ by specifying either `GT` or `GC` in the third position, respectively. Taking the example of encoding phenylalanine, `F`, the codon `TTT` encoded by ambiguous codon `DTK` is preferred over `TTC` (encoded by `DTS`) in *E. coli* by a frequency of 0.64 to 0.36. By contrast, *S. coelicolor* strongly prefers `TTC` to `TTT` to encode `F`, with frequencies of 0.97 to 0.03, respectively. A similar preference is observable in the codon usage frequencies for encoding isoleucine, `I`, in *S. coelicolor*, where `ATC` has a frequency of 0.95 compared to that of 0.03 for `ATT`. Thus, *S. coelicolor* has a strong preference for the variant codon containing `C` in the "wobble" position, and this is reflected in the scores of 0.79 for `DTS` and 0.29 for `DTK`. Organism-specific codon usage is therefore a key consideration in the design of ambiguous codons for a given host.

CodonGenie adds to a toolkit of existing software tools for ambiguous codon selection, which includes AA-Calculator (Firth & Patrick, 2008) and DYNAMCC (Halweg-Edwards et al., 2016). In contrast to AA-Calculator, CodonGenie ranks designed ambiguous codon based on their suitability for use in a given host organism. DYNAMCC also scores designed codons but offers complementary functionality to CodonGenie, as it designs sets of ambiguous codons to encode a set of amino acids with no off-target amino acid encoding and minimal redundancy. CodonGenie designs single ambiguous codons to encode a desired set of amino acids, which may also include off-target amino acids, allowing users to make a conscious trade-off between a larger library and the ease of generating such a library with a single ambiguous codon.

The above example of Table 1 illustrates a key difference between CodonGenie and DYNAMCC. Where CodonGenie will provide a list of individual ambiguous codons that will encode all desired amino acids (and potentially additional, off-target amino acids), DYNAMCC returns a single, best-scoring set of ambiguous codons that encode all desired amino acids with minimal redundancy. In the case of `F`, `I`, `L`, `M` and `V`, DYNAMCC returns the set of codons `WTT` (encoding `F` and `I` and `L`) and `VTG` (encoding `M` and `V`). The advantage of the DYNAMCC approach is in increased efficiency of the library: five DNA variants encode the five desired amino acids, while CodonGenie's solution of `DTK` or `DTS` encode six DNA variants, thus producing a larger library. The advantage of CodonGenie's solution lies in the ease in which the library can be produced with a single ambiguous codon.

CodonGenie provides a clean, intuitive web-based user interface which requires minimal user input, and which takes advantage of modern web-application development libraries such as AngularJS and Bootstrap. AngularJS (https://angularjs.org), developed and maintained by Google, provides a framework for the rapid development of modular, testable single-page web applications. Bootstrap (http://getbootstrap.com), initially developed at Twitter, provides a library of reusable user interface "widgets", such as forms, auto-fill boxes, tables, etc. Using freely available yet commercially developed libraries such as these confers a number of advantages: From a development perspective, the libraries are easy to use, are well documented and are thoroughly tested on a range of browsers (including those on mobile phones and tablets) being used perhaps billions of times a day worldwide. More importantly, the user experience is improved through use of well-developed modules that in many cases users have experienced numerous times previously in various other web applications. As a result, CodonGenie can provide a simple, easy-to-use interface that requires no documentation and can run on many platforms with the minimum of development effort.

CodonGenie is designed to follow the concept of "microservices" (Williams et al., 2016). Microservice architecture advocates the breaking down of large, monolithic applications into simple, atomic services of limited scope of functionality. By deconstructing large applications or pipelines (such as a DNA design tool) into a collection of independent units (such as a codon design module), the individual microservices can be developed, tested and deployed in isolation, increasing their reliability and reusability. CodonGenie follows this paradigm (the entire application consists of ~700 lines of code) and allows for integration into larger applications by providing a simple computer-readable RESTful web service API, as well as making itself available as a Docker container (Belmann et al, 2015; Leprevost et al., 2017), allowing users to easily redeploy their own instantiation on individual computers and services, or cloud-based platforms.

One example of the use of the CodonGenie as a microservice within a larger application is in automating the design of a synthetic DNA sequence to encode a protein sequence generated from a multiple sequence alignment. Consider a multiple sequence alignment of a hypothetic active site of an enzyme:

```
PFDMR
PIAMR
```

```
PLHLR
PMNMR
PVHMR
```

The CodonGenie webservice facilitates the writing of a simple script to automate the process of designing a synthetic DNA sequence that captures the variation encoded in this alignment. By iterating through the alignment, the set of amino acids required at each position can be collected (`{P}` for position 1, `{FILMV}` for position 2, etc.). These sets can be submitted to the CodonGenie webservice (along with a desired host organism) and a synthetic DNA sequence built up from the highest-scoring ambiguous codon returned. In practice, CodonGenie would produce the following DNA sequence for *E. coli*:

```
CCG|DTK|VMT|MTG|CGT
```

In this example, the first codon (`CCG`) is not strictly an ambiguous codon, as it contains no ambiguous nucleotides, given that a single amino acid, `P`, is required in the first position. The codon returned is the therefore the most frequent codon for encoding proline in *E. coli*. The second codon is the optimum codon for encoding `F`, `I`, `L`, `M` and `V`, as shown previously.

This example shows the benefit of offering webservice access to the CodonGenie method. While manually designing an optimised DNA sequence for a short alignment such as this is tractable, performing a similar operation on a longer alignment or a number of alignments in a manual fashion would not be feasible. Example code performing this simple operation is available (https://github.com/synbiochem/CodonGenie/blob/master/codon_genie/example/align.py), giving an indication of the ease with which CodonGenie could be incorporated into more comprehensive DNA design pipelines.

Conclusion

CodonGenie provides two simple-to-use yet valuable tools that aid the design of variant protein libraries in mutagenesis and directed evolution studies. Through both its web and web service interfaces, CodonGenie is amenable to future integration with new and existing variant library design software tools (Swainston et al, 2014). Its modular and open-source format allows for straightforward adaptation to emerging needs in the synthetic biology community, in particular the consideration of augmented genetic codes and expanded genetic alphabets (Lajoie et al, 2013; Zhang, 2017).

References

Belmann, P., Dröge, J., Bremges, A., McHardy, A.C., Sczyrba, A., Barton, M.D. (2015) Bioboxes: standardised containers for interchangeable bioinformatics software. *Gigascience*. 4: 47.

Cornish-Bowden A. (1985) Nomenclature for incompletely specified bases in nucleic acid sequences: recommendations 1984. *Nucleic Acids Res*. 13: 3021-30.

Currin, A., Swainston, N., Day, P. J., and Kell, D. B. (2015) Synthetic biology for the directed evolution of protein biocatalysts: navigating sequence space intelligently. *Chem Soc Rev*. 44: 1172-239.

Federhen, S. (2012) The NCBI Taxonomy database. *Nucleic Acids Res*. 40: D136-43.

Firth AE, Patrick WM. (2008) GLUE-IT and PEDEL-AA: new programmes for analyzing protein diversity in randomized libraries. *Nucleic Acids Res*. 36: W281-5.

Halweg-Edwards AL, Pines G, Winkler JD, Pines A, Gill RT. (2016) A Web Interface for Codon Compression. *ACS Synth Biol*. 5: 1021-3.

Hietpas, R.T., Jensen, J.D., Bolon, D.N. (2011) Experimental illumination of a fitness landscape. *Proc Natl Acad Sci U S A*. 108: 7896-901.

Jäckel, C., Kast, P., Hilvert, D. (2008) Protein design by directed evolution. *Annu Rev Biophys.* 37: 153-73.

Kille, S., Acevedo-Rocha, C.G., Parra, L.P., Zhang, Z.G., Opperman, D.J., Reetz, M.T., Acevedo, J.P. (2013) Reducing codon redundancy and screening effort of combinatorial protein libraries created by saturation mutagenesis. *ACS Synth Biol.* 2: 83-92.

Lajoie, M.J., Rovner, A.J., Goodman, D.B., Aerni, H.R., Haimovich, A.D., Kuznetsov, G., Mercer, J.A., Wang, H.H., Carr, P.A., Mosberg, J.A., Rohland, N., Schultz, P.G., Jacobson J,M., Rinehart, J., Church, G.M., Isaacs, F.J. (2013) Genomically recoded organisms expand biological functions. *Science*. 342: 357-60.

Leprevost, F.D., Grüning, B.A., Alves Aflitos, S., Röst, H.L., Uszkoreit, J., Barsnes, H., Vaudel, M., Moreno, P., Gatto, L., Weber, J., Bai, M., Jimenez, R.C., Sachsenberg, T., Pfeuffer, J., Vera Alvarez, R., Griss, J., Nesvizhskii, A.I., Perez-Riverol, Y. (2017) BioContainers: An open-source and community-driven framework for software standardization. *Bioinformatics*. doi: 10.1093/bioinformatics/btx192. [Epub ahead of print]

Lutz, S. (2010) Beyond directed evolution--semi-rational protein engineering and design. *Curr Opin Biotechnol*. 21: 734-43.

Mena, M.A., Daugherty, P.S. (2005) Automated design of degenerate codon libraries. *Protein Eng Des Sel.* 18: 559-61.

Mercer, J.A., Wang, H.H., Carr, P.A., Mosberg, J.A., Rohland, N., Schultz, P.G., Jacobson, J.M., Rinehart, J., Church, G.M., Isaacs, F.J. (2013) Genomically recoded organisms expand biological functions. *Science.* 342: 357-60.

Nakamura, Y., Gojobori, T., and Ikemura, T. (2000) Codon usage tabulated from the international DNA sequence databases: status for the year 2000. *Nucl Acids Res*. 28: 292.

Pickens, L.B., Tang, Y., and Chooi, Y.H. (2011) Metabolic engineering for the production of natural products. *Annu Rev Chem Biomol Eng*. 2: 211-36.

Pines G, Pines A, Garst AD, Zeitoun RI, Lynch SA, Gill RT. (2015) Codon compression algorithms for saturation mutagenesis. *ACS Synth Biol*. 4: 604-14.

Swainston, N., Currin, A., Day, P. J., and Kell, D. B. (2014) GeneGenie: optimized oligomer design for directed evolution. *Nucleic Acids Res*. 42: W395-400.

Williams, C.L., Sica, J.C., Killen, R.T., Balis, U.G. (2016) The growing need for microservices in bioinformatics. *J Pathol Inform*. 7: 45.

Zhang, Y., Lamb, B.M., Feldman, A.W., Zhou, A.X., Lavergne, T., Li, L., Romesberg, F.E. (2017) A semisynthetic organism engineered for the stable expansion of the genetic alphabet. *Proc Natl Acad Sci U S A*. 114: 1317-1322.

Table 1(on next page)

Comparison of codon usage frequencies for ambiguous codons encoding F, I, L, M and V in *Escherichia coli* and *Streptomyces coelicolor*.

Specific codons from two variant codons DTK and DTS are given, along with their codon usage frequency in the two organisms. For the amino acids F, I and V, there is a preference for codons with T in the third (“wobble”) position in *E. coli*, and a preference for C in the wobble position for *S. coelicolor*. This preference is reflected in the differences in scores for the ambiguous codons for the two organisms.

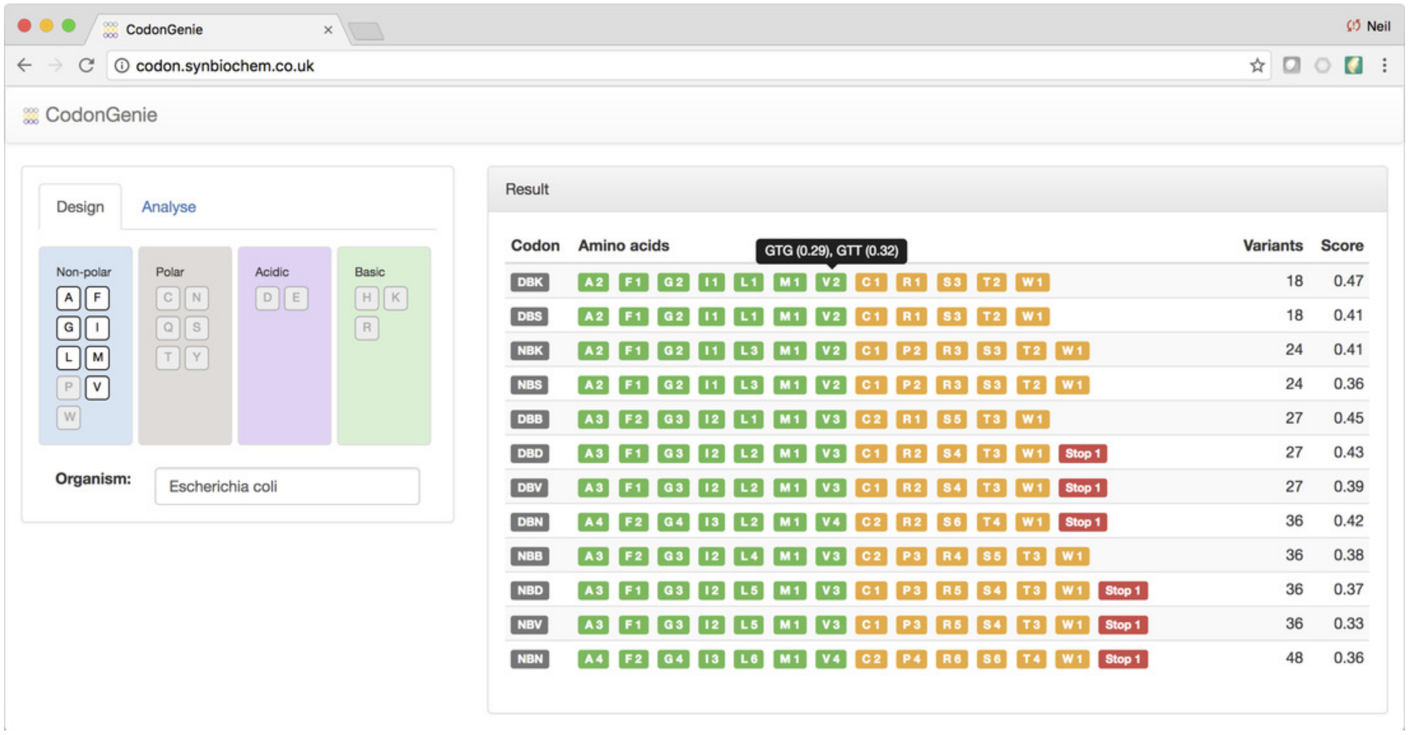
Amino Acid	**Codon**	**Ambiguous codon**	**Codon usage frequency**	
			E. coli	***S. coelicolor***
F	TTC	DTS	0.36	0.97
	TTT	DTK	0.64	0.03
I	ATC	DTS	0.31	0.95
	ATT	DTK	0.47	0.03
L	TTG	DTK and DTS	0.13	0.03
M	ATG	DTK and DTS	1.00	1.00
V	GTC	DTS	0.19	0.58
	GTG	DTK and DTS	0.29	0.36
	GTT	DTK	0.32	0.02

1

Figure 1

CodonGenie Design interface.

Users specify required amino acid combinations in the left-hand side panel. (Amino acids are grouped together in the interface in subsets of polar, non-polar, acidic and basic residues. In this example, the non-polar residues A, F, G, I, L, M and V have been selected.) Variant codons are listed in the Result panel, ordered by increasing number of Variants and decreasing codon Score (see Methods). The most specific codons are prioritised (e.g., the preferred codon in the above example, DBK, is [AGT][CGT][GT] and therefore encodes 18 DNA variants). Variant codons are shown in grey, with their encodings shown in green, orange and red for required amino acids, additional amino acids and stop codons, respectively. A given variant codon may encode an amino acid multiple times, and this is displayed in the output. For example, the preferred codon DBK encodes valine twice (with GTG and GTT) and these encodings and their codon usage frequencies may be visualised through a tooltip.



Organism: Escherichia coli

Codon	Amino acids	Variants	Score
DBK	A2 F1 G2 I1 L1 M1 V2 C1 R1 S3 T2 W1	18	0.47
DBS	A2 F1 G2 I1 L1 M1 V2 C1 R1 S3 T2 W1	18	0.41
NBK	A2 F1 G2 I1 L3 M1 V2 C1 P2 R3 S3 T2 W1	24	0.41
NBS	A2 F1 G2 I1 L3 M1 V2 C1 P2 R3 S3 T2 W1	24	0.36
DBB	A3 F2 G3 I2 L1 M1 V3 C2 R1 S5 T3 W1	27	0.45
DBD	A3 F1 G3 I2 L2 M1 V3 C1 R2 S4 T3 W1 Stop 1	27	0.43
DBV	A3 F1 G3 I2 L2 M1 V3 C1 R2 S4 T3 W1 Stop 1	27	0.39
DBN	A4 F2 G4 I3 L2 M1 V4 C2 R2 S6 T4 W1 Stop 1	36	0.42
NBB	A3 F2 G3 I2 L4 M1 V3 C2 P3 R4 S5 T3 W1	36	0.38
NBD	A3 F1 G3 I2 L5 M1 V3 C1 P3 R5 S4 T3 W1 Stop 1	36	0.37
NBV	A3 F1 G3 I2 L5 M1 V3 C1 P3 R5 S4 T3 W1 Stop 1	36	0.33
NBN	A4 F2 G4 I3 L6 M1 V4 C2 P4 R6 S6 T4 W1 Stop 1	48	0.36