ACS SyntheticBiology

Technical Note

pubs.acs.org/synthbio

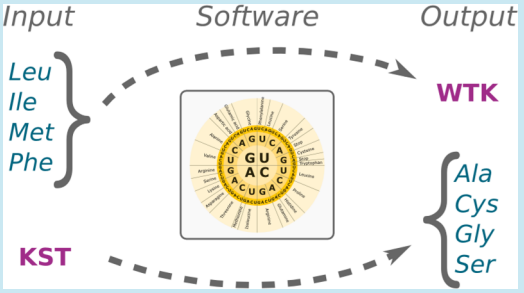
ANT: Software for Generating and Evaluating Degenerate Codons for Natural and Expanded Genetic Codes

Martin K. M. Engqvist and Jens Nielsen*

Department of Chemical & Biological Engineering, Chalmers University of Technology, Kemivägen 10, SE-412 96 Göteborg, Sweden

Supporting Information

ABSTRACT: The Ambiguous Nucleotide Tool (ANT) is a desktop application that generates and evaluates degenerate codons. Degenerate codons are used to represent DNA positions that have multiple possible nucleotide alternatives. This is useful for protein engineering and directed evolution, where primers specified with degenerate codons are used as a basis for generating libraries of protein sequences. ANT is intuitive and can be used in a graphical user interface or by interacting with the code through a defined application programming interface. ANT comes with full support for nonstandard, user-defined, or expanded genetic codes (translation tables), which is important because synthetic biology is being applied to an ever widening range of natural and engineered organisms. The Python source code for ANT is freely distributed so that it may be used without restriction, modified, and incorporated in other software or custom data pipelines.



KEYWORDS: *software, synthetic biology, protein engineering, ambiguous nucleotide, degenerate codon, mutagenesis*

For protein engineering and directed evolution, it is common to generate focused libraries of protein sequences that contain all 20 canonical amino acids, or a subset thereof, at specific positions. Such libraries are typically generated by performing PCR using DNA primers with one or more codons containing nucleotide mixes (degenerate codons). Degenerate codons are specified using the International Union of Pure and Applied Chemistry (IUPAC)-standardized ambiguous nucleotide alphabet (Supporting Information Table S1).[1] For example, the degenerate codon NNK (N = T/C/A/G; K = T/G) can be used to generate a library of sequences encoding all 20 canonical amino acids. Using the NNK degenerate codon introduces bias since it specifies 32 real codons while there are only 20 amino acids.[2] This bias can be removed by specifying a subset of real codons using combinations of degenerate codons such as the NDT + VHG + TGG codon combination[3] or the NDT + VMA + ATG + TGG combination.[4] The former reduces the number of amino acid duplications to two, and the latter, to zero.[3,4] Using such strategies to reduce redundancy is important, as the number of unique DNA sequences grows exponentially with the number of targeted sites and their respective degeneracy. Even with reduced redundancy, the number of clones that needs to be screened can be hard to manage. For example, for a library generated with the NDT + VMA + ATG + TGG codon combination at four sites, a total of 479.312 clones must be screened to reach 95% statistical library coverage.[5] To reduce the number of clones that need to be screened, one may use various structure- or sequence-based bioinformatic approaches to identify amino acid subsets to test.[6,7] Alternatively, one may use predefined degenerate codons that specify amino acid subsets with certain properties. For example, these can include amino acids that are hydrophilic (codon: VRK), hydrophobic (codon: NYC), small (codon: KST), charged (codon: RRK), or balanced in their sampling of properties (codon: NDT).[8,9] However, finding which degenerate codon to use for an *ad hoc* selection of amino acids is not trivial. Conversely, manually validating that a given degenerate codon actually encodes the specified amino acids is very time-consuming and error-prone. Tools such as DC-analyzer, MDC-analyzer, LibDesign, AA-Calculator, and GLUE-IT have been developed to assist in these processes.[4,5,8,10] While these tools are useful, they come with caveats that limit their generality. First, they do not let the user specify the genetic code used for computation. This is a limitation, as synthetic biology is increasingly applied also to organisms that do not use the standard genetic code. Second, the tools cannot handle expanded genetic codes, such as those where the UAG stop codon or the AUA isoleucine codon have been reassigned to encode unnatural amino acids (UAA).[11−15] The ability to design protein libraries making use of UAA is important, as their functional contribution can confer selective advantages.[16] Lastly, the existing tools cannot be incorporated into custom bioinformatic pipelines due to the code being unavailable or closed source. The synthetic biology and protein engineering communities would therefore benefit from an intuitive software tool for computing and evaluating degenerate codons without these limitations.

Special Issue: IWBDA 2014

Received: February 2, 2015
Published: April 22, 2015

DOI: 10.1021/acssynbio.5b00018
ACS Synth. Biol. 2015, 4, 935−938

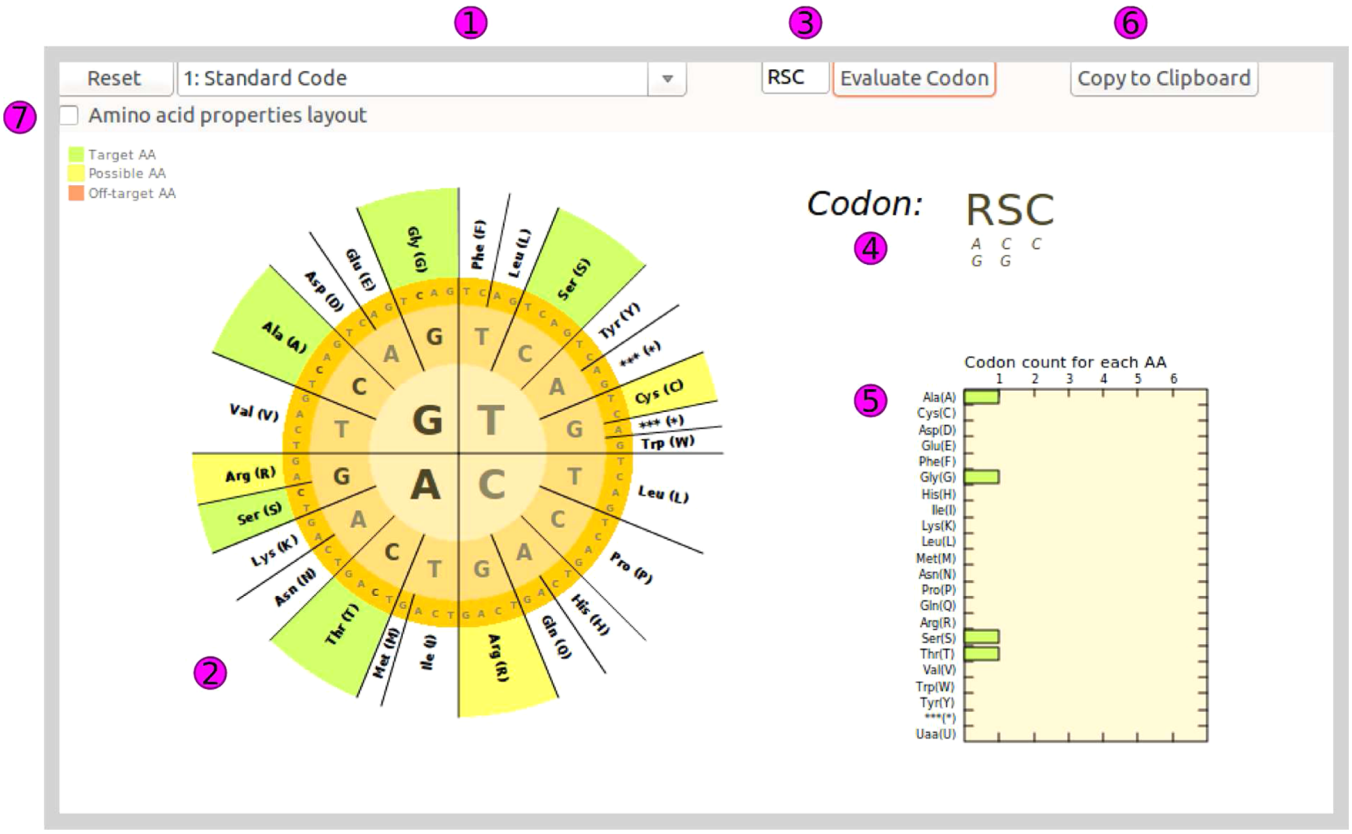


Figure 1. Overview of the ANT GUI. The different features of the GUI are indicated with magenta labels. (1) A dropdown menu for choosing the genetic code. (2) A codon circle for visualizing the amino acid selection as well as for choosing desired amino acids. (3) A text field in which a degenerate codon may be entered for evaluation. (4) The current degenerate codon is displayed in large letters, and the nucleotides specified are shown in small letters. (5) A bar graph of the codon count for each encoded amino acid. (6) Button for copying a complete report to the clipboard. (7) A checkbox that allows the user to switch between the codon wheel visualization (shown here) and the amino acid properties visualization (see Figure S1).

THE ANT APPLICATION

Here, we present the Ambiguous Nucleotide Tool (ANT), a flexible, user-friendly software tool with full support for all known genetic codes, user-defined genetic codes, and expanded genetic codes. The software runs locally on the user's computer, can compute degenerate codons for a user-defined set of amino acids, and can also evaluate which amino acids are encoded by a specific degenerate codon. For each amino acid selection, the degenerate codon with the fewest off-target amino acids and least codon redundancy is presented to the user. The software also provides a look-ahead function that highlights amino acids that may be chosen without introducing further off-target amino acids. Alternative degenerate codons can also easily be retrieved by the user. ANT is free and open source, released under the GNU General Public License, v3.0 (GPL3), such that researchers may freely study, modify, and redistribute the code. The Python source code is available at https://github.com/mengqvist/ANT.

HOW DOES ANT WORK?

ANT is a desktop application that allows the user to determine the degenerate codons for a given set of amino acids as well as to evaluate which amino acids a degenerate codon specifies. This can be done either using a Python-based command line application programming interface (API), allowing for incorporation in data pipelines, or through a point-and-click graphical user interface (GUI), allowing users with no programming experience to use the software. A user-defined genetic code that uses the UAG codon for nonstandard amino acids is provided as an example of ANT's support for expanded genetic codes.[11−14]

Parameters. In ANT, there are three parameters that may be specified: (I) the genetic code (translation table) that should be used for the computation, (II) a set of amino acids for which a degenerate codon should be computed, (III) a degenerate codon for which the encoded amino acids should be found. To compute a degenerate codon, parameters I and II need to be set. To evaluate a degenerate codon, parameters I and III need to be set. In addition to these parameters, a settings file is provided in which codons can be excluded from the computation and a user-specified codon table can be easily edited.

Working with the GUI. There are two alternate layouts for the GUI. One shows the amino acids at the periphery of a codon wheel (Figure 1), and the other indicates the physiochemical properties of each amino acid by displaying them in a series of Venn diagrams[17] (Figure S1). When working with the GUI, the genetic code appropriate for the target organism is first chosen (Figure 1, label 1). A list of genetic codes can be found at NCBI (http://www.ncbi.nlm.nih.gov/Taxonomy/Utils/wprintgc.cgi) and in Supporting Information Table S2. The user may then either start selecting amino acids by clicking on the codon wheel (Figure 1, label 2) or by specifying a degenerate codon to evaluate (Figure 1, label

DOI: 10.1021/acssynbio.5b00018
ACS Synth. Biol. 2015, 4, 935−938

Table 1. Specification of the ANT API

Command number	Input command (preceded by ">>>") with resulting output	Command description
1	`>>> import ANT` `>>> codon_object = ANT.DegenerateCodon(input=['S', 'T', 'A', 'G'], table=1)`	Generate a codon object using amino acids as an input. The variable table specifies which genetic code to use. Valid values are integers specifying the natural genetic codes: 1, 2, 3, 4, 5, 6, 9, 10, 11, 12, 13, 14, 15, 16, 21, 22, 23, 24 and 25, as well as a user-defined genetic code 1001.
2	`>>> import ANT` `>>> codon_object = ANT.DegenerateCodon(input='RSC', table=1)`	Generate a codon object using an degenerate codon as input. The variable table specifies which genetic code to use. Valid values are integers specifying the natural genetic codes: 1, 2, 3, 4, 5, 6, 9, 10, 11, 12, 13, 14, 15, 16, 21, 22, 23, 24 and 25, as well as a user-defined genetic code 1001.
3	`>>> codon_object.getTriplet()` `'RSC'`	The command getTriplet() returns the current degenerate nucleotide triplet as an upper-case string.
4	`>>> codon_object.getTarget()` `['S', 'T', 'A', 'G']`	The command getTarget() returns a list of desired amino acids in upper-case single letter code.
5	`>>> codon_object.getOffTarget()` `[]`	The command getOfftarget() returns a list of off-target amino acids (amino acids that were not chosen, yet are encoded by the degenerate codon) in upper-case single letter code.
6	`>>> codon_object.getPossible()` `['C', 'R']`	The command getPossible() returns a list of amino acids, in upper-case single letter code, which may still be chosen without adding further off-target amino acids.
7	`>>> codon_object.getTable()` `1`	The command getTable() returns the specified genetic code as an integer.
8	`>>> codon_object.getCodons()` `['ACC', 'GCC', 'AGC', 'GGC']`	The command getCodons() returns a list of regular codons, as upper-case strings, which are encoded by the degenerate codon.

9	`>>> codon_object.getCodonsPerAA()`
	`{'*': 0, 'A': 1, 'C': 0, 'E': 0, 'D': 0, 'G': 1, 'F': 0, 'I': 0, 'H': 0, 'K': 0,`
	`'M': 0, 'L': 0, 'N': 0, 'Q': 0, 'P': 0, 'S': 1, 'R': 0, 'U': 0, 'T': 1, 'W': 0,`
	`'V': 0, 'Y': 0}`
		The command getCodonsPerAA() returns a dictionary specifying how many times each amino acid is coded for by the degenerate codon. The output is a dictionary with amino acid upper-case single letter keys and integer values.

10	`>>> codon_object.getAlternatives()` `[['RSC', 'A', 'S', 'T', 'G'], ['RST', 'A', 'S', 'T', 'G']]`	The command getAlternatives() returns alternate degenerate codons which have the same number of off-target amino acids. The output is a list of lists containing triplets as an upper-case three-letter string and the amino acids encoded by that triplet as upper-case single letter code.

11	`>>> codon_object.getReport()`
	`Degenerate codon: RSC`
	`Genetic code: 1`
	`Real codons: ['ACC', 'GCC', 'AGC', 'GGC']`
	`Target amino acids: ['A', 'S', 'T', 'G']`
	`Off-target amino acids: []`
	`Amino acids that can be added w/o further off-targets: ['C', 'R']`
	`Codons for each amino acid: {'*': 0, 'A': 1, 'C': 0, 'E': 0, 'D': 0, 'G': 1,`
	`'F': 0, 'I': 0, 'H': 0, 'K': 0, 'M': 0, 'L': 0, 'N': 0, 'Q': 0, 'P': 0, 'S': 1,`
	`'R': 0, 'U': 0, 'T': 1, 'W': 0, 'V': 0, 'Y': 0}`
	`Library size (number of codons): 4`
	`Clones to screen for 95% confidence: 11`
	`Alternate codons with same number of off-target amino acids: [['RSC', 'A', 'S',`
	`'T', 'G'], ['RST', 'A', 'S', 'T', 'G']]`
		The command getReport() returns a string containing the information from all above commands. Additionally the report lists the library size and the screening burden required for 95% that all variants have been evaluated.

3). If specifying amino acids using the codon wheel, then the chosen amino acids are highlighted in green, amino acids that may still be chosen without further off-target amino acids are highlighted in yellow, and off-target amino acids are highlighted in red (Figure 1, label 2). If evaluating a codon, then the encoded amino acids are highlighted on the codon wheel in green and amino acids that may be chosen without further off-targets are highlighted in yellow. The genetic code can be changed at any time with an active amino acid selection. The degenerate codon representing the current amino acid selection is shown in large lettering, whereas the real nucleotides defined by the degenerate codon are shown in small lettering (Figure 1, label 4). The number of times each amino acid is encoded (chosen and off-target) is displayed in their respective colors in a bar graph (Figure 1, label 5). A full result report can be copied to the system clipboard (Figure 1, label 6). In addition to providing the degenerate codon, the exported information lists the chosen amino acids, off-target amino acids, amino acids that may be chosen without further off-target amino acids, the real codons defined by the degenerate codon, the genetic code used for the computation, the library size and number of clones needed to screen to reach 95% library coverage, and a list of alternate degenerate codons. The percentage library coverage returned in this report can be changed by modifying the settings file. ANT thus enables the user to very easily generate a wealth of useful information in a simple point-and-click GUI. The GUI requires a working installation of Python (https://www.python.org/) as well as wxPython (http://www.wxpython.org/).

Working with the API. For incorporation into data pipelines, ANT may also be used in a Python interpreter or imported in a Python script. A codon object is first generated by specifying amino acids and a genetic code or by specifying a degenerate codon and a genetic code (Table 1, numbers 1 and 2). Attributes of that object can then be easily retrieved and contain the same information that can be accessed through the GUI (Table 1, numbers 3−10). Finally, a complete report can be retrieved (Table 1, number 11). The API requires only a working installation of Python.

Algorithm for Generating a Degenerate Codon. When the user specifies a set of amino acids for which the degenerate codon should be found, the algorithm works as follows: Using the specified genetic code, all codons for each of those amino acids are retrieved. If any undesirable codons have been specified in the settings file, then they are removed from the list and thereby excluded from the computation. Separately, for positions 1, 2, and 3 of these codons, all ambiguous nucleotides that match at least one nucleotide for each amino acid are found. Every combination of the ambiguous nucleotides for codon positions 1, 2, and 3 is then made and represents a set of possible degenerate codons. Each degenerate codon is scored by converting back to real codons, translating to amino acids, and checking against the user-defined amino acid selection. The scoring function retains the degenerate codons with the fewest off-target amino acids (encoded amino acids that were not chosen by the user). This may be one or several codons, but they all represent the minimum number of off-target amino acids. However, these may differ in their redundancy. The scoring function therefore goes through them and selects the one that encodes the fewest number of real codons, to decrease redundancy. The resulting codon is presented to the user. To compute which other amino acids may be chosen without adding more off-target amino acids, a look-forward function is used. The look-forward function takes the current amino acid selection, goes through all of the unchosen amino acids one-by-one, and evaluates whether these lead to new off-target amino acids or not. The entire computation is finished in less than 1 s on an average laptop computer.

DOI: 10.1021/acssynbio.5b00018
ACS Synth. Biol. 2015, 4, 935−938

Algorithm for Evaluating a Codon. To evaluate a degenerate codon, the first, second, and third degenerate nucleotides are converted to their actual nucleotide counterparts. Every combination of these is then made while maintaining the position of each nucleotide. The resulting codons are translated to amino acids using the specified genetic code.

■ ASSOCIATED CONTENT

Supporting Information

Table S1: List of ambiguous nucleotides and the actual nucleotides that they specify. Table S2: List of genetic code numbers and names. Figure S1: Overview of the ANT GUI. The Supporting Information is available free of charge on the ACS Publications website at DOI: 10.1021/acssynbio.5b00018.

■ AUTHOR INFORMATION

Corresponding Author

*E-mail: nielsenj@chalmers.se.

Notes

The authors declare no competing financial interest.

■ ACKNOWLEDGMENTS

This work was supported by FORMAS and Vetenskapsrådet. We would like to thank Francesco Gatto for critical review of this manuscript.

■ REFERENCES

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

(2) Scott, J. K., and Smith, G. P. (1990) Searching for peptide ligands with an epitope library. *Science 249*, 386−90.

(3) Kille, S., Acevedo-Rocha, C. G., Parra, L. P., Zhang, Z.-G., Opperman, D. J., Reetz, M. T., and Acevedo, J. P. (2012) Reducing codon redundancy and screening effort of combinatorial protein libraries created by saturation mutagenesis. *ACS Synth. Biol. 2*, 83−92.

(4) Tang, L., Gao, H., Zhu, X., Wang, X., Zhou, M., and Jiang, R. (2012) Construction of "small-intelligent" focused mutagenesis libraries using well-designed combinatorial degenerate primers. *BioTechniques 52*, 149−58.

(5) Tang, L., Wang, X., Ru, B., Sun, H., Huang, J., and Gao, H. (2014) MDC-Analyzer: a novel degenerate primer design tool for the construction of intelligent mutagenesis libraries with contiguous sites. *BioTechniques 56*, 301−2.

(6) Jochens, H., and Bornscheuer, U. T. (2010) Natural diversity to guide focused directed evolution. *ChemBioChem 11*, 1861−6.

(7) Reetz, M. T., and Wu, S. (2008) Greatly reduced amino acid alphabets in directed evolution: making the right choice for saturation mutagenesis at homologous enzyme positions. *Chem. Commun. 43*, 5499−501.

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

(9) Balint, R. F., and Larrick, J. W. (1993) Antibody engineering by parsimonious mutagenesis. *Gene 137*, 109−18.

(10) Firth, A. E., and Patrick, W. M. (2008) GLUE-IT and PEDEL-AA: new programmes for analyzing protein diversity in randomized libraries. *Nucleic Acids Res. 36*, W281−5.

(11) 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., and Isaacs, F. J. (2013) Genomically recoded organisms expand biological functions. *Science 342*, 357−60.

(12) Johnson, D. B. F., Xu, J., Shen, Z., Takimoto, J. K., Schultz, M. D., Schmitz, R. J., Xiang, Z., Ecker, J. R., Briggs, S. P., and Wang, L. (2011) RF1 knockout allows ribosomal incorporation of unnatural amino acids at multiple sites. *Nat. Chem. Biol. 7*, 779−86.

(13) Isaacs, F. J., Carr, P. A., Wang, H. H., Lajoie, M. J., Sterling, B., Kraal, L., Tolonen, A. C., Gianoulis, T. A., Goodman, D. B., Reppas, N. B., Emig, C. J., Bang, D., Hwang, S. J., Jewett, M. C., Jacobson, J. M., and Church, G. M. (2011) Precise manipulation of chromosomes in vivo enables genome-wide codon replacement. *Science 333*, 348−53.

(14) Mukai, T., Hayashi, A., Iraha, F., Sato, A., Ohtake, K., Yokoyama, S., and Sakamoto, K. (2010) Codon reassignment in the *Escherichia coli* genetic code. *Nucleic Acids Res. 38*, 8188−95.

(15) Bohlke, N., and Budisa, N. (2014) Sense codon emancipation for proteome-wide incorporation of noncanonical amino acids: rare isoleucine codon AUA as a target for genetic code expansion. *FEMS Microbiol. Lett. 351*, 133−44.

(16) Liu, C. C., Mack, A. V., Tsao, M.-L., Mills, J. H., Lee, H. S., Choe, H., Farzan, M., Schultz, P. G., and Smider, V. V. (2008) Protein evolution with an expanded genetic code. *Proc. Natl. Acad. Sci. U.S.A. 105*, 17688−93.

(17) Taylor, W. R. (1986) The classification of amino acid conservation. *J. Theor. Biol. 119*, 205−218.

DOI: 10.1021/acssynbio.5b00018
ACS Synth. Biol. 2015, 4, 935−938