CoLiDe instruction manual

Vyacheslav Tretyachenko, Vaclav Voracek

May 31, 2020

Contents

1 Introduction

CoLiDe (degenerate codon optimization tool) serves for automatic design of randomized protein libraries with given amino acid distribution. Inputs of the program involve amino acid distribution, length of resulting proteins in the library and level of degeneracy. Optional parameters are organism codon bias, removed triplets and codon reassignment. Program outputs a set of degenerate or spiked codons which provide a protein library with desired characteristics. Principle of method lies in stochastic genetic algorithm, therefore solutions for approximations of a given distribution can differ between different runs.

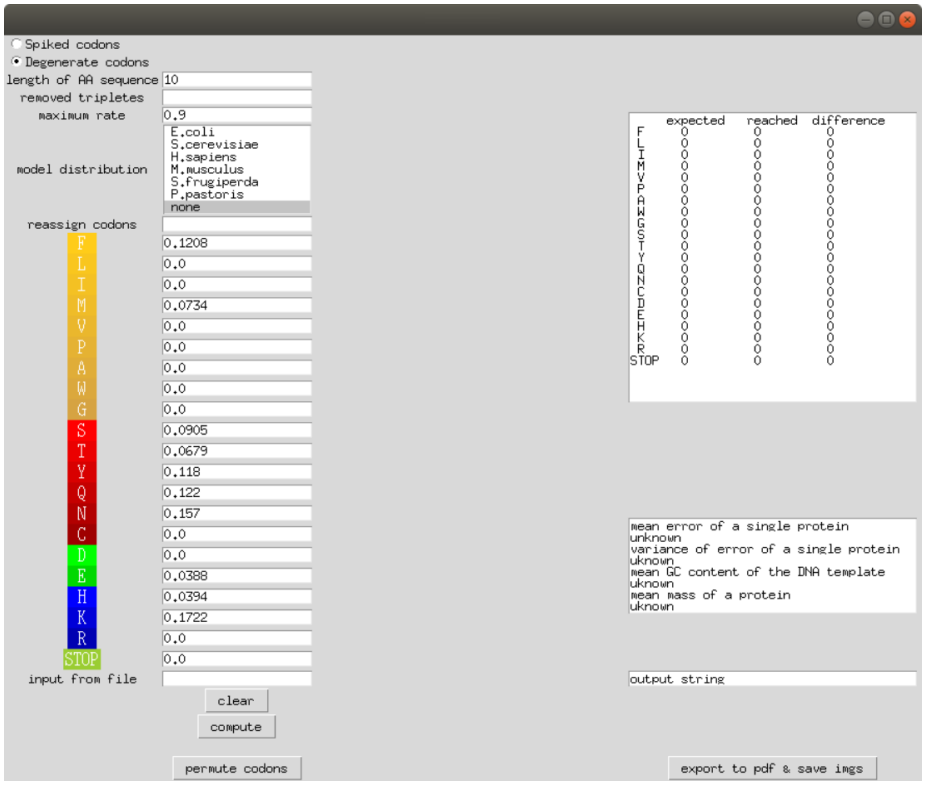

Figure 1: Main window of degenerate codon optimization tool – CoLiDe

2 Installation

CoLiDe prerequisites are Python 3 and supplementary packages listed in requirements.txt. In case that you do not have Python installed we recommend to download Miniconda from https://docs.conda.io/en/latest/miniconda.html which will install Python 3 and few additional useful packages. After Miniconda installation:

1. open Anaconda prompt and run the command (without quotation marks) "cd xxx", where xxx is the path to the directory with the CoLiDe.

2. On first use run command (without quotation marks) "conda install --file requirements.txt"

3. run the command "python gui.py"

On a few MacOS machines script was not able to load. This is due to serious bugs with Apple supplied Python. In case that happens, it is recommended to install the newest version of python and start script with command "pythonw gui.py". (https://www.python.org/download/mac/tcltk/)

If Python 3 is already installed on your computer or/and you do not want to install Miniconda, use "pip install" command to install all prerequisite packages in requirements.txt file. After the installation change your working directory in terminal by command "cd xxx", where xxx is the path to the directory with CoLiDe and run CoLiDe with "python gui.py" command.

Although CoLiDe was tested on Linux distribution Ubuntu 18.04, other operating systems (Windows, MacOS) should be compatible with fully python-based script.

3 Setting up

3.1 Mandatory input

Obligatory inputs for CoLiDe are length of the library proteins and amino acid distribution. These attributes can be defined in "length of AA sequence" and specific amino acid brackets. Amino acid ratios do not need to be normalized to sum to 100 or 1. Ratios of amino acids (e.g A 1, T 2, S 2) will be normalized to fractions of 1 automatically (therefore A 0.2, T 0.4, S 0.4). An alternative way for input is provided by "input from file" (Section 3.2).

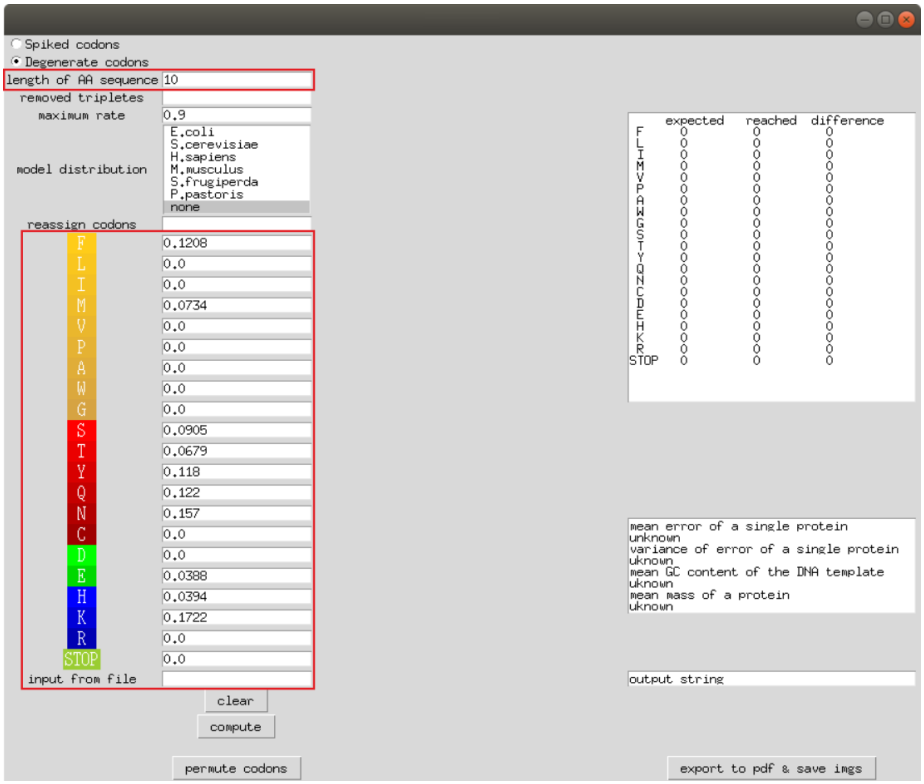

Figure 2: Most important variables for resultant degenerate libraries - length in amino acids and amino acid composition. Both variables can be entered manually or using file input

3.2 Input from file

Input file must be ".txt" file. It needs to contain amino acid distribution, which will be further normalized. If an amino acid is not present in input file, it will not appear in the output at all. Further items are optional, if they will not be present in the input file, they will be read from the GUI.

Here we provide an example content of such input file.

```
F=1
L=2
A=1
T=5
Q=2.5
N=0.3
D=0.12
H= 5
K=5
length=11
model_distribution=E.coli
maximum_rate=0.95
removed_triplets=AAA GGG
reassigned_codons=ACT 2, GTG 3
spiked
```

Frequency of amino acid is coded by its one-letter abbreviation, followed by the equality sign and a numerical value. First letter of the line has to be the amino-acid name and the second is the equality sign. The numerical value can be inserted in arbitrary form. Note that "." is used as decimal separator.

Further lines of the example input specify optional parameters for CoLiDe. Each parameter must be defined exactly and must be followed by a "=". Only exceptions are "spiked" and "degenerated" parameters, which enable/disable the usage of spiked codons without any other specification.

Other parameters are followed by the same format of input as in the GUI, with the exception of model_distribution, which contains a name of one of distributions in the GUI.

If a line differs in format from those described above (even a superfluous space counts!), then it is ignored and may be used for comments.

The order of lines does not matter.

3.3 Degenerate codon type

CoLiDe offers approximation via degenerate (default) and spiked codons. Optimization with spiked codons might take longer time to converge, but offer higher precision in shorter library lengths, difficult amino acid combinations or in cases where more solutions with higher degeneracy are required. In longer templates (> 15 amino acids) the difference in precision between degenerate and spiked solutions become negligible.

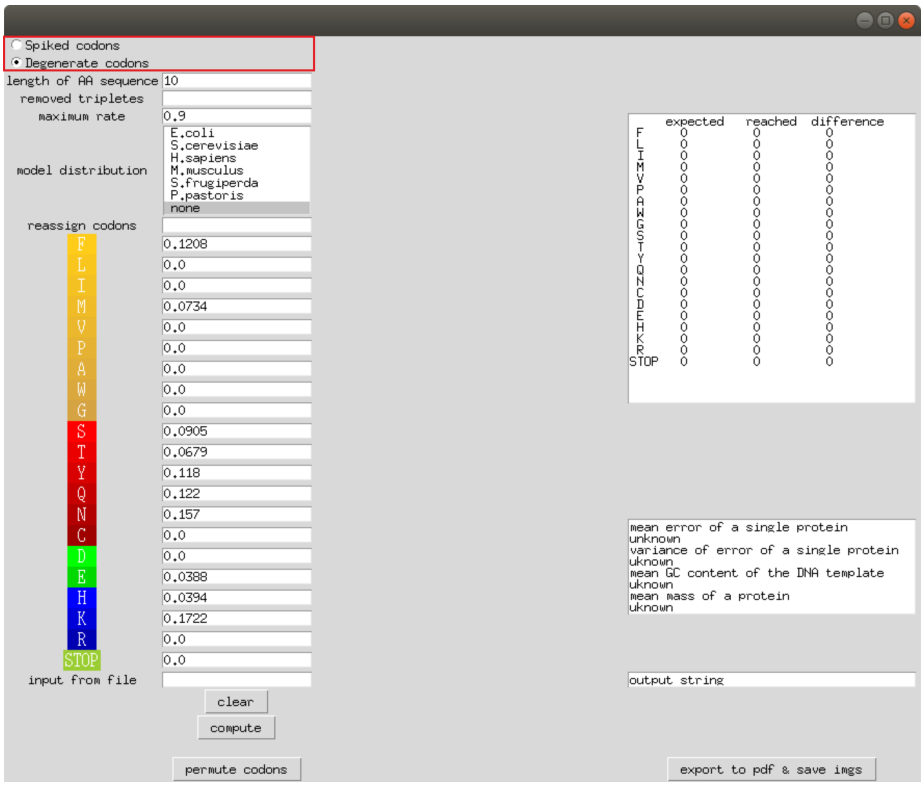

Figure 3: Standard degenerate codons with equimolar base ratios and spiked codons with variable base ratios can be chosen for distribution approximation

3.4 Codon removal and reassignment

Specific triplets can be removed by specifying triplets in space separated format into the “removed triplets” bracket (for example ”AAA ACT GTG”). CoLiDe will output solutions containing only those degenerate codons which do not code removed triplets. Another option is codon reassignment. If defined in the “reassign codons” in format ”XXX number” (e.g. ATG 0.3), codon will be treated as another amino acid (and no longer as methionine) with appropriate ratio (0.3) and will be included into the degenerate output. There may be multiple instanes of such reassigned codons. In the input, they are separated by commas (for instance ”ACT 0.074, GTG 0.111”)

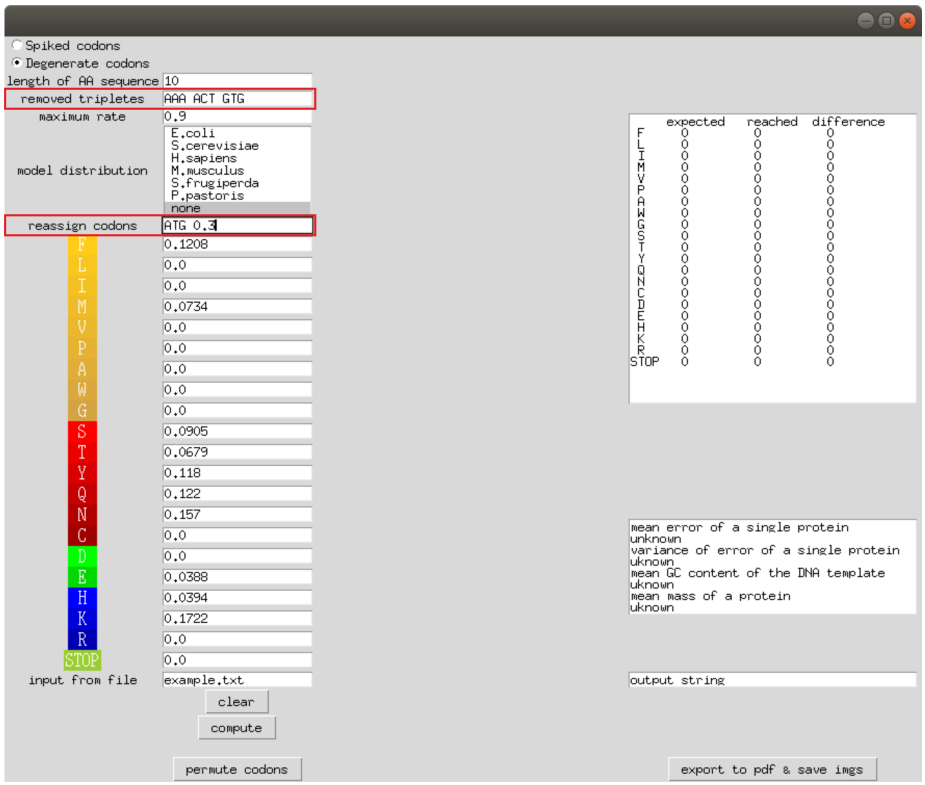

Figure 4: Certain triplets can be removed from final degenerate solution or reassigned to encode a different amino acid

3.5 Organism codon bias

Output can be optimized according to organism codon bias, solution will be enriched in codons preferred by the selected model organism.

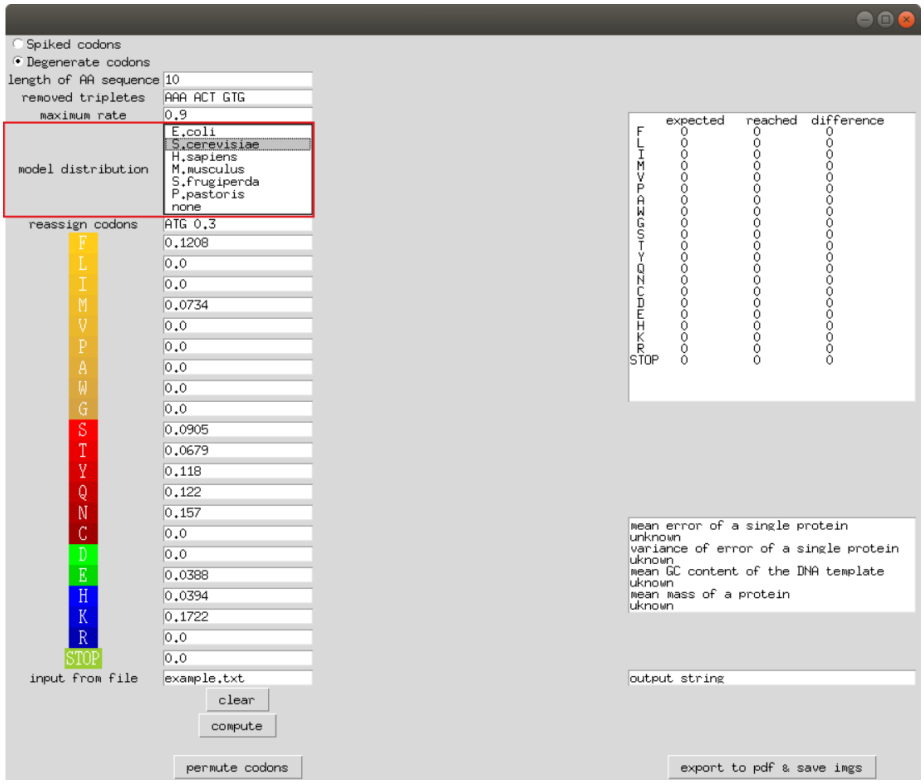

Figure 5: CoLiDe can start approximation from codon optimized seed template. This option increases the probability of organism specific triplets enrichment in final degenerate solution

3.6 Example output

With default settings and distribution, solution can be generated by "compute" button. Calculation can take several minutes depending on template library length, type of codons and specific amino acid alphabet.

Output provides essential statistics on amino acid approximation and differences from input, variance protein amino acid content in the designed library, GC content of designed DNA library and mean molecular weight of protein product. Degenerate codons are represented by pie charts where slices are contents of hydrophobic (shades of yellow), polar (reds), negatively (greens) and positively charged (blues) amino acids. Numbers above the pie chart represent positions of degenerate codons in the template. Codon positions are arbitrary and can be shuffled by the "permute codons" button.

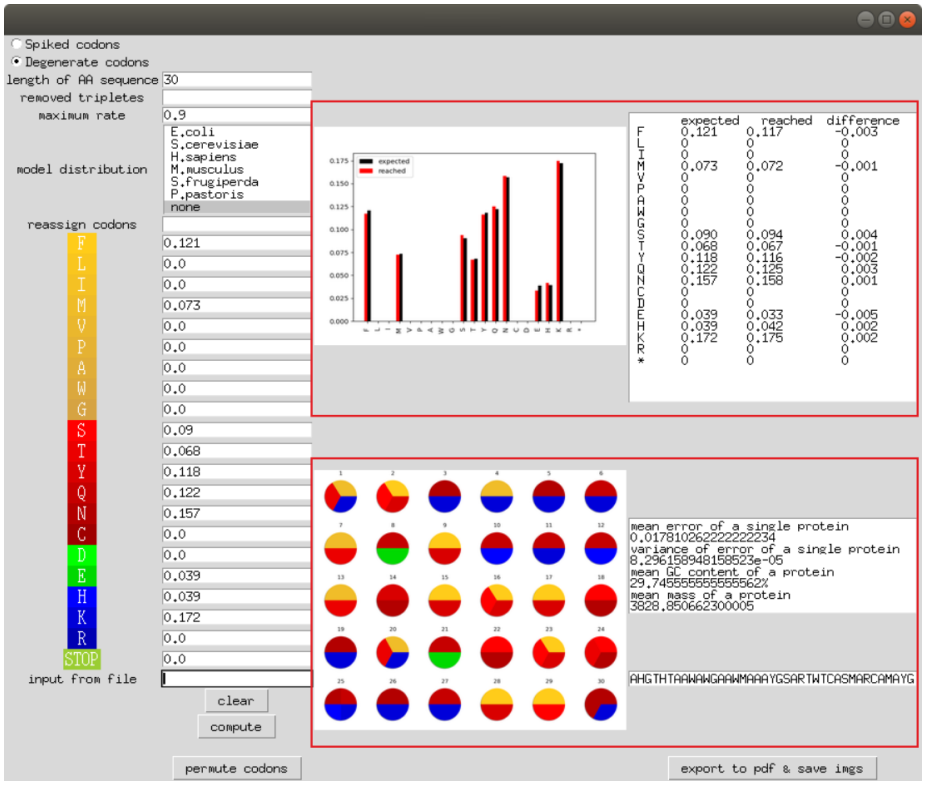

Figure 6: Example output of default settings and amino acid distribution on 30 amino acid library template

3.7 Degeneracy of solution

Degeneracy of solution can be tuned by "maximum rate" parameter. Rate represents maximum ratio of amino acid representable by degenerate codon (e.g. codon RGN codes for two arginines, two serines and four glycines, so ratios of R, S and G are 0.25, 0.25 and 0.5 respectively - thus codon RGN will never be included in solution if "maximum rate" was set on 0.2).

Here is the example solution where the maximum ratio was limited to 0.4 (e.g. maximum representation of a single amino acid by degenerate codon is 40 %). Pie charts provide a visual proof of increased degeneracy compared to previous example output with default rate of 0.9. In some cases, higher degeneracy comes with a price of lower precision as shown here. This occurs when input amino acid codons are not very similar and increased degeneracy significantly limits the codon pool for approximation. If precision is important, we recommend to increase maximum amino acid rates or use spiked codons.

Default ratio 0.9 ensures that only degenerate codons are included in the solution, maximum ratio 1 will include non-degenerate codons and lower ratios will use more degenerate codons accordingly.

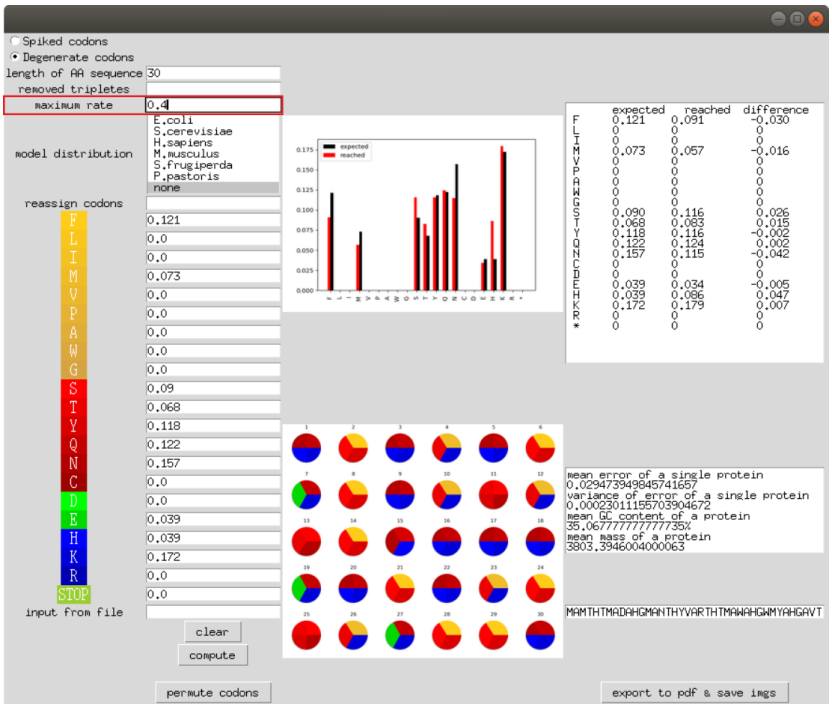

Figure 7: Degeneracy of solution can be fine-tuned, using lower values of rate parameter for higher degeneracy up to 1 for lowest degeneracy where algorithm is allowed to use non-degenerate codons for approximation

3.8 Spiked codons

Precision of solution can be improved by inclusion of spiked codons (e.g. codons with variable nucleotide ratios) into the design scheme. Here we present how spiked codons improve the precision of previously shown solution with increased degeneracy. However, not all commercial oligonucleotide synthesis providers are able to follow the spiked codon format and feasibility of the solution as a synthesis template should be consulted individually.

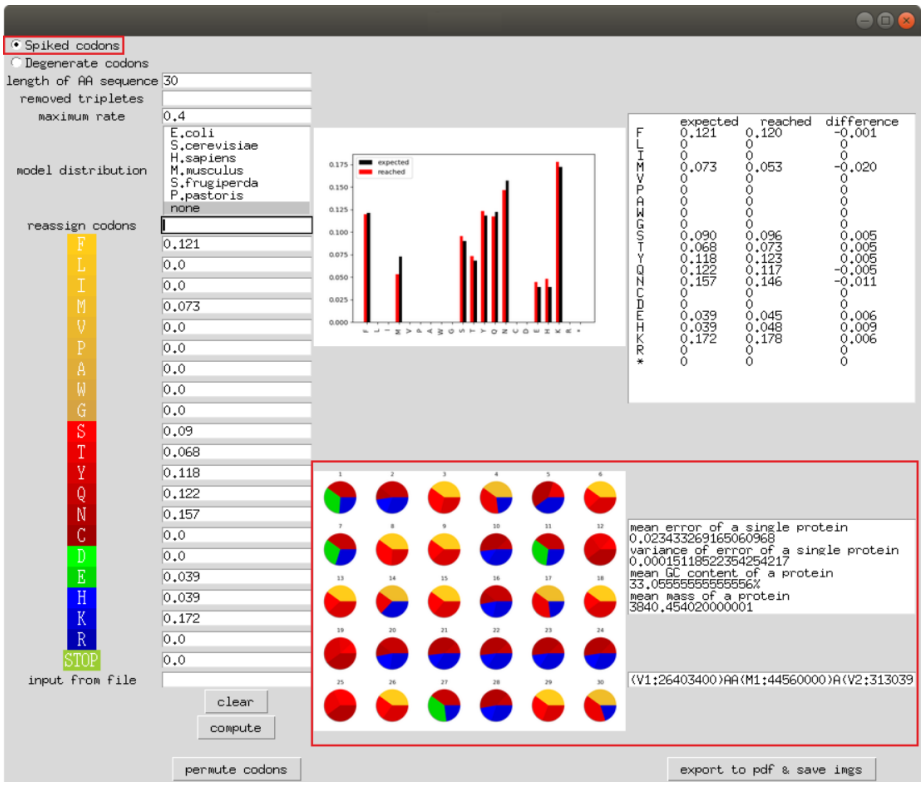

Figure 8: Utilization of spiked codons can bring higher precision. Disadvantage is necessity of individual consultation with oligonucleotide synthesis company

3.9 PDF report

All statistics shown in the main window are exported to pdf file in the CoLiDe directory.