GENTANGLE manual

v1.3

April 24, 2024

Abstract

This document describes the technical details of GENTANGLE (Gene Tuples ArraNGed in overLapping Elements), a high-performance containerized pipeline for the computational design of two overlapping genes translated in different reading frames of the genome. Details about the different parts of the pipeline and how to use them to obtain and analyze entangled variants are provided. The GENTANGLE wiki may have more updated documentation and the GENTANGLE website may store a more updated version of this manual.

GENTANGLE can be used to design and test gene entanglements for microbial engineering projects using arbitrary sets of user specified gene pairs.

Contents

1 Introduction

GENTANGLE pipeline is built around CAMEOX (CAMEOs eXtended), an enhanced, parallelized version of CAMEOS (Constraining Adaptive Mutations using Engineered Overlapping Sequences) that we release as the computational core of GENTANGLE. GENTANGLE aims at automating the entire process of gene entanglement, by enhancing its computational scalability, improving software portability, and adding new functional and analysis capabilities.

1.1 Repositories

GENTANGLE: The GENTANGLE repository wiki contains updated instructions on how to use the container and the different components of software and data. The GENTANGLE source code and its submodules are freely available on GitHub at https://github.com/BiosecSFA/gentangle.

DATANGLE: The DATANGLE (DATA for genTANGLE) repository freely available at https://github.com/BiosecSFA/datangle on GitHub contains related data and results that are useful for running and testing GENTANGLE and easily reproducing results.

CAMEOX: CAMEOX (CAMEOs eXtended) is a Julia code that is the computational core of GENTANGLE pipeline. CAMEOX is a parallelized, enhanced version of CAMEOS. It is a git submodule of GENTANGLE and is freely available on GitHub at https://github.com/BiosecSFA/CAMEOX.

pyCAMEOX: pyCAMEOX python module is a git submodule of GENTANGLE with auxiliary code both for the upstream and downstream (sub)pipelines. It is freely available on GitHub at https://github.com/BiosecSFA/pycameox.

Ortho: This OrthoDB-based MSA generation pipeline is a git submodule of GENTANGLE. This pipeline is released as a potential alternative to the UniRef-based MSA generation that is integrated on GENTANGLE, but is not currently supported. It is freely available on GitHub at https://github.com/BiosecSFA/ortho

1.2 License

2 CAMEOX

2.1 Overview

This repository contains code and data related to CAMEOX (CAMEOs eXtended), a parallelized extension of CAMEOS (Constraining Adaptive Mutations using Engineered Overlapping Sequences) developed by LLNL (Lawrence Livermore National Laboratory). The original CAMEOS software was developed by Tom Blazejewski at Wang Lab (Columbia University). CAMEOX is the computational core of the GENTANGLE pipeline for automated design of gene entanglements.

2.2 Installation

2.2.1 As part of GENTANGLE

The recommended installation method is as part of the GENTANGLE pipeline by cloning the GENTANGLE repository or, even better, by downloading the Singularity container as this eases the process of setting all the many requirements of CAMEOX, and also the DATANGLE repository to provide data examples and templates. Please see this link for details on these approaches.

2.2.2 Only CAMEOX source code

```
git clone https://github.com/BiosecSFA/cameox.git
```

2.3 CAMEOX improvements over CAMEOS

The main improvements in CAMEOX relative to CAMEOS are:

2.3.1 Performance and Scalability

- **Parallelization:** CAMEOS optimization is not parallelized, while CAMEOX main optimization loop is parallelized by using shared-memory threads with optimized garbage collection on Julia. This improvement allows for a larger number of variants to be evaluated in parallel.
- **Dynamic stopping criteria:** CAMEOS works with a fix, static number of iterations, while that number is variable in CAMEOX below a given maximum: the main optimization loop is automatically stopped by a dynamic condition based on the relative number of variants evolving per iteration. This enhancement allows for a larger number of variants to be evaluated by limiting redundant calculations on an exponentially growing number of variants that stop evolving over time.

- **Redundancy reduction:** CAMEOS has only the "cull" mechanism using predefined NPLL (negative-pseudo-log-likelihood, aka anti-pseudo-log-likelihood or APLL) limits, while CAMEOX is skipping variants in each cycle accounting for their evolution over the iterations. This addition allows for a larger number of variants to be evaluated by limiting the redundancy of scoring the same variant.
- **Numerical stability:** CAMEOX resolves several bugs in CAMEOS, thereby greatly improving the numerical stability and robustness of the code. CAMEOX is able to run with an order of magnitude more sequences and for longer duration (more iterations for optimization) without crashing thus allowing the generation of more candidate solutions.

2.3.2 Flexibility and Customization

- **Generalized codon optimization:** CAMEOS codon optimization is hardwired for *E. coli*, while CAMEOX includes a generalized embedded codon optimization by reading from an external database. This CAMEOX extension allows users to design genes for other microorganisms with different codon usage tables.
- **Adjustable mutagenesis parameters:** CAMEOS uses hardwired mutagenesis parameters, while CAMEOX give users the ability to modify the mutagenesis parameters used in the optimization algorithm. This addition enable users to more aggressively mutate each variant and potentially explore a more diverse sequence space for candidate solutions.
- **Detached data directory:** CAMEOS requires data and code in the same pre-established directory structure, while in CAMEOX the data directory can be detached from code directory and locations are flexible. This improvement allows for greater installation flexibility, since the data directories can become large and may need to be stored in a separate location from the software.
- **Customizable optimization weights:** CAMEOS has internal hardwired values for the NPLL optimization, while CAMEOX exposes a user option with several common choices offered (see details below in subsection about PLL weights for optimization). This improvement allows the user to customize their design criteria to weight the fitness importance of one gene over the other as needed.

2.3.3 Output and Analysis

- **Reference sequence NPLL values:** CAMEOX outputs NPLL values calculated for reference sequences (usually WT, wild type), which are used in downstream normalization to enable comparisons between different runs. This new feature allows fitness scores to be evaluated relative to the WT and can enable comparison between the output of different models used for the genes. CAMEOS does not provide these values.
- **Comprehensive metadata:** CAMEOX generates comprehensive metadata to enable downstream management of multiple pairs and runs. This expansion helps with running a larger scale search over multiple gene pairs. CAMEOS is not providing metadata files.

- **Pre-MRF optimization variants:** CAMEOX can provide complete results for the variants post-HMM optimization but pre-MRF optimization to allow for checks or alternative optimization methods. This new feature is useful for evaluating differences between the HMM and MRF variants and potentially assess different optimization techniques. CAMEOS lacks this capability.

2.3.4 Usability

- **Entanglement frame awareness:** CAMEOX is aware of the working entanglement frame related to the longer gene (see subsection below with details). This enhancement allows for user to set and confirm if the shorter gene is embedded in the second or third reading frame related to the longer gene. CAMEOS does not account for this data.

2.4 CAMEOX parameters

2.4.1 Format of the parameter file

CAMEOX improvements over CAMEOS have required some changes in the TSV input/parameters file from column 7 regarding CAMEOS. Each line in the file should now have the following columns:

1. Output dir: relative base directory where the output directory will be created.
2. Mark gene name: gene ID string for 'mark' gene; needed as a key for looking up some values associated with genes in files.
3. Deg gene name: gene ID string for the corresponding 'deg' gene.
4. Mark JLD file: relative path to mark gene JLD file.
5. Deg JLD file: relative path to mark gene JLD file.
6. Mark HMM file: relative path to mark gene HMM directory and `.hmm` file.
7. Deg HMM file: relative path to deg gene HMM directory and `.hmm` file.
8. Population size: number of seeds that will enter the optimization loop, i.e. number of individual HMM solutions to greedily optimize.
9. Frame (placeholder): p1/p2/p3, but the entanglement frame depends on the order of the genes in the input (see subsection below for details).
10. Relative change threshold: minimum threshold for the relative number of variants changing, used for setting a dynamic limit on the number of iterations; typical value for standard CAMEOX runs is 0, or very close.
11. Host taxid: NCBI Taxonomic ID for the host of the entanglement, used by the host generalization subsystem (the default value is 562, for *E. coli*; see subsection below for details).
12. Pseudolikelihoods weights for optimization choice, which should be one of the next options: `equal`, `rand`, `close2mark`, `close2deg` (see subsection below for details).

2.4.2 Example

Example of a single-line CAMEOX parameter file with *Pseudomonas protegens* Pf-5 (NCBI taxid: 220664) as target host for the entanglements:

```
output/ aroB_pf5 infA_pf5 jlds/aroB_pf5.jld jlds/infA_pf5.jld hmms/aroB_pf5.hmm
[continuation line] hmms/infA_pf5.hmm 20000 p1 0 220664
```

2.4.3 Entanglement frame

As indicated above in the input format, the frame parameter in the parameter/input file is a placeholder, both in CAMEOS and CAMEOX. The effective way to select the entanglement frame is via the order of the genes in the input. Using CAMEOS terminology, typically, the "mark" gene is the shorter gene and the "deg" gene is the longer gene. By inverting that order, the effective frame of entanglement regarding the longer gene is changed. CAMEOX is aware of the working entanglement frame and outputs that information at the start of any run to clarify the actual entanglement frame:

```
Processing entanglement [shorter_prot]⇄[longer_prot] in frame [real_frame]
```

where `[real_frame]` can be either `5'3'F2` or `5'3'F3`.

2.4.4 Host selection

As previously mentioned, CAMEOS codon optimization is hardwired for *E. coli*, while CAMEOX includes a generalized embedded codon optimization by reading from an external database. This database is composed by one TSV file for each organism used as host for the entanglements. Each filename follows the format `CUT_{taxid}.tsv`, where CUT stands for Codon Usage Table and `taxid` is the taxonomic identifier for the organism in the NCBI Taxonomy database. Each TSV file needs two columns: 'codon' for the codons and 'freq' for the frequencies. As an example, please see *Pseudomonas protegens* Pf-5 (NCBI taxid: 220664) CUT file. The DATANGLE repository also contains the *E. coli* (NCBI taxid: 562) CUT file direcly usable by CAMEOX.

In case that additional hosts are targeted, a quick method to get the CUT is to consult an online CoCoPUTs service, retrieve the CUT for the desired host with NCBI taxonomic identifier `hostTaxId`, and save it with the described format in the file `CUT_{hostTaxId}.tsv`, which should be placed in the root of CAMEOX data directory.

2.4.5 Pseudolikelihood weights for optimization

As indicated above in the format of the CAMEOX parameter file, the last parameter indicates the pseudolikelihood (PLL) weights for optimization. Before the MRF optimization (main optimization loop), each gene of each pair of

HMM seeds is assigned a weight. Within a pair, the weights sum `1.0` and indicate the relative importance of each gene PLL (as calculated by the respective MRF models) for the total pair score. The options for this parameter are the following:

- `equal`: The weight will be always equal for both genes (`0.5`). So, there is no optimization preference for one over the other regarding the PLL.
- `rand`: For each pair of HMM seed in the population of variants, the weight for one of the genes is randomly obtained from a uniform prob distribution between 0 and 1 so the weight of the other is taken to that both sum `1.0`. Since is very difficult to known a priori the relative importance of both genes for a successful entanglement, this is the preferred choice when working with a large number of variants to be able to better explore the space of solutions and generate a workable Pareto's front.
- `close2mark`: The weight will be always `1.0` for the mark gene and `0.0` for the deg gene, thus optimizing only for the mark gene. This may be useful in extreme entanglement cases where the relative importance of the mark gene is orders of magnitude above the one of the deg gene.
- `close2deg`: The weight will be always `1.0` for the deg gene and `0.0` for the mark gene, thus optimizing only for the deg gene. This may be useful in extreme entanglement cases where the relative importance of the deg gene is orders of magnitude above the one of the mark gene.

2.4.6 Notes

- For mark and deg genes names, if you have used the upstream pipeline, please use the same strings here.
- You will need to use the same mark and deg genes names in the downstream pipeline.

2.5 Further documentation

- Please see the GENTANGLE wiki for useful documentation about the overall pipeline and the Singularity container that include CAMEOX (recommended method for running the code).
- The original CAMEOS manual may still be useful.
- For related code, data, documentation, and notebooks specific to Livermore Computing (LC) you can take a look at this repo if you have access to LC.

2.6 License

3 DATANGLE

The DATANGLE (DATA for genTANGLE) repository contains data for GENTANGLE protein entanglement pipeline.

3.1 Overview

The purpose of DATANGLE is providing both a prepared directory structure and pre-populated example data for GENTANGLE. This data can be used in examples to quickly run the different entry points on GENTANGLE's Singularity container (please see details here). It is important to bind the local directory containing the data with a specific directory within the container (see details below). In the examples below in this wiki page, we will assume that you have installed DATANGLE in `/path/datangle` where `/path` is whatever path you may have chosen.

3.2 Getting the DATANGLE repo

The DATANGLE repo provides the necessary files and directory structure to run the GENTANGLE pipeline. You can download it using git (if you don't have git installed in your system you can download from the official git website. To get DATANGLE, you just need to clone the repository:

```
git clone https://github.com/BiosecSFA/datangle.git
```

NOTE: If you are not entangling your own genes of interest and are testing the CAMEOX pipeline with the two genes the original CAMEOS paper presented (infA and aroB), you can use pre-built MSA and trained HMM/MRF models suited for CAMEOX, all available in the DATANGLE repository. You will not need to download the protein database or specify custom input gene sequences. You can generate entanglement proposals using the DATANGLE pre-build and pre-computed CAMEOX required files for this example. (see Getting DATANGLE example files)

3.3 Getting git LFS DATANGLE example files

A couple of data files are larger than the size that the standard git repositories are usually able to manage and an extension to git called git-lfs is required. This extension is automatically installed with git in recent versions of git. Depending on the version of git and the git-lfs extension that you have in your system, you may need to run additional commands to download the pre-computed example files for infA and aroB. This step is not required if you are entangling your own genes of interest and have downloaded the protein database and specified custom input gene sequences.

3.3.1 Checking that you have the git extension manager available

Try `git lfs version`. If you get an error, then you will need to install git-lfs using the instructions on the git lfs installation page or, even better, upgrade your git software to the latest version, that you can download from the official git website.

3.3.2 Install the lfs extension

Just run the next git command:

```
git lfs install
```

3.3.3 Pull the data

In the same directory where you have cloned the DATANGLE repository issue the next command:

```
git lfs pull
```

IMPORTANT: If you get an error from GitHub refusing to download the files related to the bandwidth quota of the LFS service, please contact the authors as soon as possible using your preferred method —opening an issue in the repository is a totally valid choice. Contrary to other providers, GitHub charges for the bandwidth of the LFS service even for public/educational/research projects by using data packages. As of April 2024, git LFS receives 1 GiB of free storage and 1 GiB a month of free bandwidth on GitHub. With our LFS files reaching 2/3 of the free quota for the storage, the bandwidth is quickly exhausted when the users clone the repository and pull the files from the GitHub LFS server. It is easy that the free bandwidth (or successive data packages) get exhausted before we can purchase an upgrade. If you find this situation, please reach out to us at your early convenience.

3.3.4 Check the data

Check the size of the next large data files:

- `/path/datangle/jlds/aroB_pf5_uref100.jld` ~ 451 MB
- `/path/datangle/jlds/infA_pf5_uref100.jld` ~ 17.5 MB
- `/path/datangle/msas/aroB_pf5_uref100.sto` ~ 203 MB

4 GENTANGLE basics

In this section you can find useful links and updated documentation about GENTANGLE (Gene Tuples ArraNGed in overLapping Elements), a high performance containerized pipeline for the computational design of two overlapping genes translated in different reading frames of the genome. GENTANGLE can be used to design and test gene entanglements for microbial engineering projects using arbitrary sets of user specified gene pairs.

CAMEOX (CAMEOs eXtended), a parallelized and enhanced version of CAMEOS, is at the core of the GENTANGLE pipeline. Details about the different parts of the pipeline and how to use them to obtain and analyze entangled variants are provided in the different pages of this wiki. For licensing information please see the repository.

4.1 Requirements

As with many pieces of software, the requirement of the GENTANGLE pipeline depends on the use case. If the target are generating tens of thousands of entangled variants for many proteins-pairs in a reasonable amount of time, an HPC platform is almost unavoidable. For less demanding use cases, a multi-core Linux machine should be enough.

GENTANGLE's Singularity container comes in two flavors, the standard (CPU-only) and the GPU, which provides explicit support for Nvidia/CUDA.

4.1.1 Minimum configuration

- Operating system: GNU/Linux (native or via virtual machine)
- Storage: 3 GB when using pre-built MSA for the gene pairs
- Container: only `gentangle` Singularity image

4.1.2 Preferred configuration

- Operating system: GNU/Linux
- CPU: multi-core processor
- Storage: 200 GB of free space if building MSA from scratch, 20 GB of free space otherwise but when using the `gentangle_gpu` Singularity image.
- GPU: if GPU acceleration is desired when training the MRF models, a CUDA-compatible GPU card is required and then the `gentangle_gpu` Singularity image should be used.

4.1.3 Alternative configurations

While we don't support these configurations, the software can be run on Mac and Windows platforms, concretely, by using a Linux virtual machine (VM) able to run the Singularity container. The CUDA-compatible GPU card is recommended for accelerating the step of training some of the models when entangling new protein pairs, but it is not mandatory. The size of the storage required may be drastically reduced if the user skips the step of GENTANGLE that generates MSAs for the entanglement pair and provides their own alignment.

4.2 Installation (recommended procedure)

The preferred GENTANGLE installation and use is via the GENTANGLE Singularity container.

4.2.1 Prerequisite: Singularity software

The GENTANGLE Singularity container encapsulates all the many required dependencies. Of course, you still need Singularity software to be able to run the container in your machine. For running the GENTANGLE Singularity container, we recommend a recent Singularity version, at least 3.11.1.

Singularity is usually installed on HPC systems, but an administrator may install it from source or from RPM/DEB packages. Additional methods are detailed in the installation section of the Admin Guide and quick installation in the quick installation steps of the User Guide.

4.2.2 GENTANGLE Singularity image download

You can download the image in two different versions:

- Default/CPU-only (~1.5GB), from either:
  - Sylabs public Singularity Cloud Library at `https://cloud.sylabs.io/library/khyox/gentangle/gentangle.sif`. The container should be accessible with the following command: `singularity pull gentangle.sif library://khyox/gentangle/gentangle.sif`
  - LLNL ftp server at `ftp://gdo-bioinformatics.ucllnl.org/gentangle/gentangle.sif` using your preferred software supporting anonymous ftp download.
- GPU-enabled (~15GB), from LLNL ftp server at `ftp://gdo-bioinformatics.ucllnl.org/gentangle/gentangle_gpu.sif` using your preferred software supporting anonymous ftp download.

The GPU-enabled version has been built with explicit support for Nvidia/CUDA GPU platforms. While the training of MRF models can be done with just CPUs, it can be substantially accelerated when using a GPU. Please see below details about how to accomplish this in the **mrftrain** step of [[End-to-end example|Running-the-pipeline#End-to-end-example-for-an-entanglement]].

4.2.3 GENTANGLE Singularity image inspection (optional)

You may inspect the metadata of the container with the next command:

```
singularity inspect gentangle.sif
```

Container version should be 1.0.0 or higher and build date April 2024 or later.

4.2.4 GENTANGLE Singularity image verification (optional)

You may verify the PGP signature of the container with the next command:

```
singularity verify gentangle.sif
```

You should see something like the next output:

```
INFO:    Verifying image with PGP key material
[REMOTE]  Signing entity: Jose Manuel Marti (email1) <email2>
[REMOTE]  Fingerprint: 33E31B147E97CF7451D93A3C586F7D642537E104
Objects verified:
ID  |GROUP   |LINK    |TYPE
------------------------------------------------
1   |1       |NONE    |Def.FILE
2   |1       |NONE    |JSON.Generic
3   |1       |NONE    |JSON.Generic
4   |1       |NONE    |FS
INFO:    Verified signature(s) from image 'gentangle.sif'
```

If the verification fails, check your network and try again. If it keeps failing and your network is OK, try to download again the container. If the verification keeps failing or fails because mismatching signature, please don't continue and contact the authors.

4.3 Installation (alternative procedure)

An unsupported alternative for GENTANGLE installation is cloning the repository and manually installing all the (many) dependencies. The GENTANGLE git repository includes several submodules, so please use the `--recursive` argument when cloning the repository:

```
git clone --recursive https://github.com/BiosecSFA/gentangle.git
```

4.4 Next steps

0. Preparing the required input of the pipeline. In addition, we also recommend that you check the documentation of DATANGLE regarding prepared directory structure and pre-populated example data for GENTANGLE.
1. Running the different steps of the pipeline via the GENTANGLE Singularity container.
2. Understanding the output of the pipeline.

5 Required input data for running GENTANGLE

The prerequisites for running the GENTANGLE pipeline from scratch for a gene pair are:

1. DATANGLE (recommended!)
2. A protein database for homologous search (unless the user provides their own alignments).
3. Custom input gene sequences (both amino acids and nucleotides sequences in fasta format)
4. Parameters for the entanglement (can be easily adapted from the example on DATANGLE) and, potentially, a CUT (Codon Usage Table) for the targeted host (DATANGLE-provided CUTs can be used directly or as templates). These are direct requirements of CAMEOX, GENTANGLE's core.

5.1 DATANGLE

While this is not technically a requirement, we strongly recommend to clone the DATANGLE repo as it provides:

1. A fully compatible data directory structure with GENTANGLE,
2. Example files with GENTANGLE inputs, which are very useful as templates for different gene pairs or parameters,
3. Example files with GENTANGLE output that are useful to learn in advance about the typical results.
4. All the intermediate files to be able to independently test any step of the GENTANGLE pipeline with the Singularity container by using the example entanglement of infA and aroB (these two genes are entangled in the original CAMEOS paper).

Please see details and documentation on the DATANGLE wiki page.

5.2 Protein database

5.2.1 Do I need to download the database?

A protein database is required for finding sequences homologous to your genes of interest in order to build the protein multiple sequence alignments required by CAMEOX. However, if you already have MSAs for the genes to entangle or are testing the pipeline using our provided examples, you can skip this download.

The GENTANGLE container provides an app (the first step of the pipeline) to create MSAs from scratch using an updated local protein database version (a UniProt's UniRef database recommended), which enables local generation of alignments. While we support and recommend this approach, the user is free to generate MSAs using any other procedure and then skip that step of the pipeline. If you are entangling your own genes and don't have pre-built alignments for them, you'll need to download a protein database, so please proceed to the next subsection.

5.2.2 Getting the protein database

You will need a protein database in fasta format. We recommend that you download Uniprot's UniRef100 or UniRef90 databases and place it/them at the `/path/datangle/dbs` directory. They are large databases so you will need some free space for them as indicated in the requirements: `uniref90.fasta` for example is 86 GB as of April 2024. The databases can be download from here.

We suggest using UniRef90 for genes that are frequently sequenced and have many orthologs. If the resulting alignments have a limited number of orthologs, and especially if they have less orthologs than the length of the reference sequence in amino acids, we recommend using UniRef100 instead. If there is doubt we suggest starting with the UniRef100, understanding that there may be some sequence redundancy in the retrieved sequences. The sequence redundancy can be managed using the included `ppmsa` app of the GENTANGLE container for curating the multiple sequence alignments with the `--seqid_filter` option to further remove redundant sequences (see more details in the step 2 of the pipeline).

You may use different protein databases in fasta format. Despite the size, we recommend using UniProt's UniRef for its good coverage of a variety of proteins, but some users may prefer to use local metagenomic databases, which are even larger than the recommended ones, but may potentially provide a larger training set.

5.3 Custom input gene sequences

0. **Gene name**: Once you assign a label (geneLabel1) to your reference protein, you must consistently keep that naming throughout the pipeline, so it should be carefully selected in this "step 0". As an example, we use the structure `gene_organism_database` in our examples, where geneLabel1 is named `infA_pf5_uref100`.

1. **Fasta sequences**: In order to specify the two gene you'd like to entangle using CAMEOX, you need to create fasta files containing their sequences in both amino acids for the proteins and nucleotides for the coding region sequences (CDS). `geneLabel1.refseq.fa` is a standard fasta file containing the protein sequence of geneLabel1, with header `>geneLabel1`, and `geneLabel1.cds.fa` is a standard fasta file containing the nucleotide CDS of geneLabel1:

    1. Copy the protein sequence to a new subdirectory within the `msas` data directory under `/path/datangle/` (needed for all the upstream GENTANGLE pipeline):

        ```
        mkdir /path/datangle/msas/geneLabel1/
        cp geneLabel1.refseq.fa /path/datangle/msas/geneLabel1/
        ```

    2. Add the protein sequence to the `proteins.fasta` file in the root of the data directory (needed for CAMEOX step in GENTANGLE):

```
cat geneLabel1.refseq.fa >> /path/datangle/proteins.fasta
```

3. Add the CDS to the `cds.fasta` file in the root of the data directory (needed for CAMEOX step in GENTANGLE):

```
cat geneLabel1.cds.fa >> /path/datangle/cds.fasta
```

2. **Repeat for the other gene of the pair**: You will specify the second gene with the exact same procedure of steps `0` and `1` above. Again, you can replace geneLabel2 with any unique name of your gene, but the label need to be consistent in the path, filename, and fasta headers.

3. **Proceed with GENTANGLE pipeline**: Once the custom input genes are setup you can then follow steps outlined the end-to-end example to build the multiple sequence alignment, which are steps 1 and 2 of the GENTANGLE pipeline; train the protein fitness models, which are steps 3 and 4 of GENTANGLE; and then summarize all the required files in step 5. Once steps 1-5 are complete, the entanglement search is run as step 6 of GENTANGLE (CAMEOX) and evaluation and analysis of its output is done in steps 7 and 8 of the pipeline. However, before running step 6 (CAMEOX) you will need to set the required parameters for the CAMEOX execution (see next section).

5.4 CAMEOX parameters and CUTs

CAMEOX parameters and CUTs for different hosts are described in detail in the CAMEOX readme. As recommended in that document, in both cases, for any customization, we recommend starting from:

- The parameter file in DATANGLE
- CUT files in the DATANGLE repository, such as the *E. coli* (NCBI taxid: 562) CUT file

6 Understanding GENTANGLE Output

This wiki page provides a comprehensive guide to the various output files and sampling strategies generated by the GENTANGLE pipeline, enabling users to effectively utilize the results in gene entanglement design. Understanding these outputs is crucial for making informed decisions and optimizing your research workflow around GENTANGLE. Here you will find details about the directory structure, different file formats (including interactive plots, tabular data, and serialized dataframes), and the various sampling strategies available for downselecting candidate solutions.

6.1 Output Directory Structure

GENTANGLE organizes its output within a dedicated `outputs` subdirectory. Within this directory, a new subfolder is created for each pair of entangled genes, named in the format `{geneLabel1}_{geneLabel2}`. To be informative, our gene labels typically include information about the reference sequence strain and the database used to build the multiple sequence alignments (MSAs) for each gene, for instance `gene_organism_database` in the case of the genes in the `infA_pf4_uref100`⇄`aroB_pf4_uref100` entanglement.

6.2 Output Files

The final solutions are automatically placed in different files in the output directory (e.g., `/path/datangle/output`).

6.2.1 Summary File

- **File format:** `summary_{RUN_ID}.csv`
- **Description:** This file provides a comprehensive overview of all generated solutions for the entanglement of `{geneLabel1}` and `{geneLabel2}`. The `RUN_ID` is a unique identifier automatically assigned during the `cameox` step of the pipeline, ensuring traceability.

6.2.2 Sampling Strategies and Output Files

GENTANGLE often generates a large number of candidate solutions, exceeding the capacity for experimental testing, since it is recommended to generate as exhaustive a search as possible to find the best performing solutions. To facilitate downselection, the pipeline offers four distinct sampling strategies, each producing a separate output file:

1. Pareto Frontier

- **Output file format:** `pareto_ERP_CAMEOX_{geneLabel1}_{geneLabel2}_pairs.csv`

- **Description:** This strategy selects the top N solutions along the Pareto frontier, alternating solutions with optimal trade-offs between the best fitness scores of `{geneLabel1}` and `{geneLabel2}`. This sampling strategy samples strictly from the best scoring solutions and should output reliable entanglement solutions when accurate fitness models were used, so it is ideal when high confidence exists in the accuracy of the fitness models.

2. Multiplicity

- **Output file format:** `multy_ERP_CAMEOX_{geneLabel1}_{geneLabel2}_pairs.csv`
- **Description:** This downsampling strategy selects the top N solutions by the number of times that different HMM seeds converged to the same solution after the MRF optimization for the entanglement of `{geneLabel1}` and `{geneLabel2}`. This redundancy can be observed within a run and between different runs of CAMEOX, and this criterion accounts for both. This method offers an alternative sampling strategy to the Pareto frontier by identifying solutions that the pipeline consistently converges upon.

3. Overdensity

- **Output file format:** `overden_ERP_CAMEOX_{geneLabel1}_{geneLabel2}_pairs.csv`
- **Description:** This strategy bins solutions based on their score distribution and samples N solutions evenly from the top M most populated but most separate regions for the entanglement of `{geneLabel1}` and `{geneLabel2}`. It is useful in early discovery phases where model accuracy may be uncertain, ensuring a set of samples predictive to have an optimized range of diverse fitness values.

4. Random

- **Output file format:** `random_ERP_CAMEOX_{geneLabel1}_{geneLabel2}_pairs.csv`
- **Description:** This strategy randomly samples variants from the entire solution space for the entanglement of `{geneLabel1}` and `{geneLabel2}`. Similar to overdensity sampling, it is beneficial in early stages or when exploring a wide range of possibilities without relying heavily on model scores.

6.2.3 Entanglement Relative Position (ERP)

Definition and examples **ERP** (Entanglement Relative Position) refers to the relative position of the shorter gene within the longer gene, expressed as a decimal value with four digits for precision. For example:

- A shorter gene embedded in a 10000 nucleotide length gene at position 1 would be at relative position 0.0001 and thus represented as ERP0001 (Entanglement Relative Position 0.1‰ or 0.01%).

- A shorter gene embedded in a 10000 nucleotide length gene at position 10 would be at relative position 0.001 and thus represented as ERP0010 (Entanglement Relative Position 1‰ or 0.1%).
- A shorter gene embedded in a 1000 nucleotide length gene at position 1 would also be at relative position 0.001 and thus represented as ERP0010 (Entanglement Relative Position 1‰ or 0.1%).
- A shorter gene embedded in a 1000 nucleotide length gene at position 900 would also be at relative position 0.9 and thus represented as ERP9000 (Entanglement Relative Position 900‰ or 90%).

Rationale for ERP Decimal Precision The choice of four decimal places for ERP values strikes a balance between precision, practicality, and user-friendliness. It allows for unique positioning within protein sequences up to 10,000 amino acids, which covers a significant majority of known proteins. While some redundancy may occur for proteins exceeding 10,000 amino acids, the frequency of such proteins is relatively low, making the risk of uncertainty very low. Using five or more decimal places would likely be unnecessarily complex and confusing for users, especially considering the low frequency of very large proteins and its practicality in this context of gene entanglement.

6.3 Interactive Output Visualization

- **File format:** `iplot*.html`
- **Description:** GENTANGLE generates interactive HTML files that allow users to visually explore the distribution of solutions and the impact of different sampling strategies for the entanglement of `{geneLabel1}` and `{geneLabel2}`. Double-clicking on each sampling strategy within the interactive plot reveals the selected candidate solutions; per each ERP, the options are: `pareto_ERP`, `multy_ERP`, `overden_ERP`, and `random_ERP`.
- **Example**: Example of Interactive Output. Try selecting the Pareto option for the two different ERP values to see how the model clearly favors one entanglement position over the other. Note in the example, the term **~ratio** (similarity ratio) refers to the fraction of matching amino acids between the design protein and the reference (wild type) protein, so the averaged ~ratio is the similarity average ratio considering both proteins in the entanglement pair.

6.4 Additional Data Files

Beyond the CSV and HTML files, GENTANGLE produces additional data files containing scores, parameters, and statistics. These files offer deeper insights into the solution space and can be used for further analysis, visualization, or downstream applications such as machine learning or custom filtering. They can be analyzed using the provided example Jupyter notebook:

- **Serialized Pandas DataFrames:**

  - **File format:** `CAMEOX_{geneLabel1}_{geneLabel2}_pairs_*.pkl.bz2`
  - **Description:** These compressed files contain Pandas DataFrames storing detailed information about the generated solutions for the entanglement of `{geneLabel1}` and `{geneLabel2}`, including scores, sequences, and other relevant data.
- **Metadata:**
  - **File format:** `CAMEOX_{geneLabel1}_{geneLabel2}_metadata.csv`
  - **Description:** This file provides metadata associated with the CAMEOX run for the entanglement of `{geneLabel1}` and `{geneLabel2}`, such as parameters used and runtime statistics.

6.5 Choosing a Sampling Strategy

The choice of sampling strategy depends on the confidence in the fitness models and the stage of the research project. When model accuracy is high, prioritizing the Pareto frontier is recommended as it focuses on solutions with the best trade-offs between gene fitness scores. In early stages or when working with noisy or less reliable models, incorporating multiplicity, overdensity, or random sampling can be beneficial. These strategies provide a broader exploration of the solution space and can be used to collect additional data for further model refinement in subsequent cycles. Ultimately, GENTANGLE offers all four sampling options to allow users to experiment and find the most suitable approach for their specific research goals and model confidence.

7 Running the pipeline via the apps of the Singularity container

Singularity is a software used to bring containers and reproducibility to scientific computing and the high performance computing (HPC) world. Another common software used for containerization is Docker. We created a container image with all software tools used in this pipeline pre-installed. The software tools can be run either as apps using the commands in this tutorial, or interactively from a container created from the singularity image.

7.1 Working with the different apps

7.1.1 Overview

The different apps are different points of entry into the container that follow the SCIF standard. You can list the apps with the following command:

```
singularity inspect --list-apps gentangle.sif
```

And you may get a list of apps in alphabetical order similar to the next one, depending on your version of the container:

- **analysis** - Collate output from CAMEOX run and generate ranked candidate solutions and interactive visualization of solutions.
- **cameox** – Run CAMEOX, using HMM and MRF optimization to generate entangled variants of a gene pair.
- ccmpred - Auxiliary entry point to CCMpred software (not intended to be used directly as part of the pipeline, use mftrain instead).
- hhsuite - Auxiliary entry point to HH-suite software (not intended to be used directly as part of the pipeline, use ppmsa instead).
- **hmmer** – Generate initial alignments with target protein searched against a protein database such as UniRef100 using HMMER suite.
- **hmmtrain** – Train hidden Markov protein models (HMMs) for initial seed sequences to be refined by MRF model.
- julia – Auxiliary entry point to Julia framework (not intended to be used directly as part of the pipeline).
- **mrftrain** – Train Markov random field (MRF) protein fitness model.
- **outparser** – Parse results from the output dataset of CAMEOX runs.
- **ppmsa** – Generate curated multiple sequence alignment for designated protein from initial alignment obtained with hhmer app.
- python – Auxiliary entry point to Python framework (not intended to be used directly as part of the pipeline).
- **summarize** – Retrieve the scoring parameters from training models to be used in downstream CAMEOX runs.

You can retrieve usage and help information of each one of the apps. For instance, for details about the `outparse` app that parses the output of CAMEOX you can execute the next line:

```
singularity run-help --app outparser gentangle.sif
```

And you should get something like the next output:

```
Gentangle outparser step (downstream CAMEOX)
Arguments: <runid> [<mark_gene> [<deg_gene> [<frame>
[<fasta> [<CAMEOX_output_path>]]]]]
    with defaults:
        <mark_gene> = aroB_pf5_uref100
        <deg_gene> = infA_pf5_uref100
        <frame> = p1
        <fasta> = all
        <CAMEOX_output_path> = $SCIF_APPDATA_cameox/output
Example: singularity run --cleanenv --app outparser --mount type=bind, \
source=/home/user/datangle,dst=/scif/data/cameox gentangle.sif saRGGj2F \
aroB_pf5_uref90 infA_pf5_uref90 p1
```

7.1.2 Binding the data directory

It is essential to connect the local directory with input data (and outputs later), such as DATANGLE, to a specific directory inside the container. The bind is done with the next arguments in the singularity command line: `--mount type=bind,source=/path/datangle,dst=/scif/data/cameox`. Alternatively, if you do not have a path containing ':', you can use the next syntax instead of the `--mount` option: `--bind /path/datangle:/scif/data/cameox`. This alternative syntax may be the only choice to set this in the command line if you are using an older version of Singularity.

7.2 End-to-end example for an entanglement

The eight steps detailed below correspond to eight different container apps (entry points to the container using the Scientific Filesystem standard), with names highlighted in bold. These steps match the enumerated in the paper's Figure 1A (top subfigure). We provide examples for each one of the steps using data in DATANGLE for a infA⇄aroB entanglement, but the user can easily use the examples as a template for their own gene pair.

0. **Preparatory step**: A "step 0" is needed to provide some required input data for the pipeline (please see [[input wiki pagelinput]] for further details):

- Some files required by CAMEOX (this is already provided with DATANGLE): parameters for the runs and the sequences of the proteins in fasta format for both amino acids and nucleotides.
- In the user plans to generate their own alignments from scratch (step 1 of the pipeline), a protein DB in fasta format is required for the search of homologs using HMM models in order to build the MSAs. This can even be a potentially huge DB such as one from metagenomic analysis, but we recommend at least a version of UniProt UniRef DBs. The DB should be placed on `/path/datangle/dbs`; for instance, the path to the UniRef100 DB fasta file would be `/path/datangle/dbs/uniref100.fasta`. We don't provide these DBs as they are large files that are updated often —we recommend to use a current version.

7.2.1 Per individual gene to be entangled

---

1. **hhmer**: Generation of the initial alignment with jackhmmer (HMMER) for the infA and aroB protein of the *Pseudomonas* PF5 host based on UniProt's UniRef100 database:

   ```
   singularity run --cleanenv --app hmmer --mount type=bind,source=/path/datangle,\
   dst=/scif/data/cameox gentangle.sif infA_pf5_uref100 \
   /scif/data/cameox/dbs/uniref100.fasta

   singularity run --cleanenv --app hmmer --mount type=bind,source=/path/datangle,\
   dst=/scif/data/cameox gentangle.sif aroB_pf5_uref100 \
   /scif/data/cameox/dbs/uniref100.fasta
   ```

   The output of this step should contain:

   ```
   /path/datatangle/msas/infA_pf5_uref100/targets.tblout
   /path/datatangle/msas/infA_pf5_uref100/infA_pf5_uref100.sto
   /path/datatangle/msas/aroB_pf5_uref100/targets.tblout
   /path/datatangle/msas/aroB_pf5_uref100/aroB_pf5_uref100.sto
   ```

   and a number of `.sto` and `.hmm` files generated during the run as results of intermediate iterations.

---

2. **ppmsa**: Postprocessing and curating the MSA based on EVcouplings for the same protein. We show here two examples of this step:

   - No sequence identity filtering in order to have a larger quantity of natural sequences:

```
singularity run --cleanenv --app ppmsa --mount type=bind,source=/path/datangle,\
dst=/scif/data/cameox gentangle.sif infA_pf5_uref100 \
/scif/data/cameox/msas/infA_pf5_uref100

singularity run --cleanenv --app ppmsa --mount type=bind,source=/path/datangle,\
dst=/scif/data/cameox gentangle.sif aroB_pf5_uref100 \
/scif/data/cameox/msas/aroB_pf5_uref100
```

- Filter sequences in the alignment with 95% or more sequence identity to others to reduce possible redundancy:

```
singularity run --cleanenv --app ppmsa --mount type=bind,source=/path/datangle,\
dst=/scif/data/cameox gentangle.sif --seqid_filter 95 infA_pf5_uref100 \
/scif/data/cameox/msas/infA_pf5_uref100

singularity run --cleanenv --app ppmsa --mount type=bind,source=/path/datangle,\
dst=/scif/data/cameox gentangle.sif --seqid_filter 95 aroB_pf5_uref100 \
/scif/data/cameox/msas/aroB_pf5_uref100
```

---

3. **hmmtrain**: Train of HMM model with the results of the previous example:

```
singularity run --cleanenv --app hmmtrain --mount type=bind,source=/path/datangle,\
dst=/scif/data/cameox gentangle.sif infA_pf5_uref100

singularity run --cleanenv --app hmmtrain --mount type=bind,source=/path/datangle,\
dst=/scif/data/cameox gentangle.sif aroB_pf5_uref100
```

---

4. **mrftrain**: Train of MRF model with the results of the alignment postprocessing step:

   - Using only CPUs (standard Singularity container `gentangle.sif`), with 256 threads:

```
singularity run --cleanenv --env THREADS=256 --app mrftrain --mount type=bind,\
source=/path/datangle,dst=/scif/data/cameox gentangle.sif infA_pf5_uref100

singularity run --cleanenv --env THREADS=256 --app mrftrain --mount type=bind,\
source=/path/datangle,dst=/scif/data/cameox gentangle.sif aroB_pf5_uref100
```

- Using GPU acceleration
  (GPU-enabled Singularity container `gentangle_gpu.sif`):
  - Use defaults:

```
singularity run --cleanenv --nv --app mrftrain --mount type=bind,\
source=/path/datangle, dst=/scif/data/cameox \
gentangle_gpu.sif infA_pf5_uref100

singularity run --cleanenv --nv --app mrftrain --mount type=bind,\
source=/path/datangle, dst=/scif/data/cameox \
gentangle_gpu.sif aroB_pf5_uref100
```

  - Set a specific I/O directory, different number of iterations (100 is the default), and a different GPU device (0 is the default):

```
singularity run --cleanenv --nv --app mrftrain --mount type=bind,\
source=/path/datangle,dst=/scif/data/cameox gentangle_gpu.sif \
infA_pf5_uref100 /scif/data/cameox/msas/infA_pf5_uref100 20 1

singularity run --cleanenv --nv --app mrftrain --mount type=bind,\
source=/path/datangle, dst=/scif/data/cameox gentangle_gpu.sif \
aroB_pf5_uref100 /scif/data/cameox/msas/aroB_pf5_uref100 40 2
```

---

5. **summarize**: Pre-CAMEOX summarization step to retrieve the required energies and pseudolikelihoods of the sequences in the alignment, warning this step can take on the order of 35 minutes to run (for each gene):

```
singularity run --cleanenv --app summarize --mount type=bind,source=/path/datangle,\
dst=/scif/data/cameox gentangle.sif infA_pf5_uref100

singularity run --cleanenv --app summarize --mount type=bind,source=/path/datangle,\
dst=/scif/data/cameox gentangle.sif aroB_pf5_uref100
```

7.2.2 Per entanglement (involving the couple of genes to be entangled)

NOTE: The next step assumes that you have run the previous steps for the two genes to be entangled, which are `infA_pf5_uref100` and `aroB_pf5_uref100` in this example that follows DATANGLE data.

6. **cameox**: CAMEOX run with all the prerequisites ready using the example parameter file `aroB_infA_pf5_uref100_debug.txt` from DATANGLE and 256 threads on a shared-memory node:

```
singularity run --cleanenv --env THREADS=256 --app cameox --mount type=bind,\
source=/path/datangle,dst=/scif/data/cameox gentangle.sif \
/scif/data/cameox/paramfiles/aroB_infA_pf5_uref100_debug.txt -n rand -l rand -g -g
```

7. **outparser**: Post-CAMEOX step to retrieve detailed results for the CAMEOX execution with barcode `3Tgcn6kD` —the barcodes are generated automatically per run in order to avoid overlap in filenames and content. Please replace `3Tgcn6kD` with the barcode of your own CAMEOX run.

```
singularity run --cleanenv --app outparser --mount type=bind,source=/path/datangle,\
dst=/scif/data/cameox gentangle.sif 3Tgcn6kD aroB_pf5_uref100 infA_pf5_uref100
```

   The automatically generated barcode can be found in:

   (a) messages to the console and log output of CAMEOX:
   `/path/datangle/output/geneLabel1_geneLabel2_p1/log_<barcode>.csv`

   (b) the filename of output and log files such as:
   `/path/datangle/output/geneLabel1_geneLabel2_p1/saved_pop_<barcode>.csv`

   (c) the contents of the metadata file:
   `/path/datangle/output/geneLabel1_geneLabel2_p1/`
   `/CAMEOX_geneLabel1_geneLabel2_metadata.csv`, which contains the metadata for all the runs for the same gene pair and frame.

   Results of this step will be placed in:
   `/path/datangle/output/aroB_pf5_uref100_infA_pf5_uref100_p1` in our example. Key output files are: `summary_<barcode>.csv` and `variants_<barcode>.fasta`, which contain the model scores for each entanglement design in a csv file and a fasta file with the raw sequences and informative headers.

   In addition, there are equivalent files for the sequences produced from the first stage of entanglement using just the HMM models, previous to any MRF-based optimization. This allows for the use of your own custom models for refining sequence proposals or study the net trajectory of the variants in the solution space due to the optimization. These additional files are: `sum_hmm_<barcode>.csv` and `vars_hmm_<barcode>.fasta`. This step may take some time depending on the number of variants of the CAMEOX run.

IMPORTANT: You can skip this 7th step if you are interested in the final analysis of all the CAMEOX runs for an entanglement (next/final 8th step), as the analysis code will automatically call this 7th step (outparser) iteratively as needed for all the barcodes corresponding to the CAMEOX runs for the specified entanglement.

---

8. **analysis**: A comprehensive analysis of the CAMEOX run(s) for a given entanglement, including interactive visualization of the proposed and selected sequences, can be automatically generated using the example Jupyter notebook:

   ```
   singularity run --cleanenv --app analysis --mount type=bind,source=/path/datangle,
   dst=/scif/data/cameox gentangle.sif aroB_pf5_uref100 infA_pf5_uref100
   ```

   Final results of this step will be placed in `/path/datangle/output/aroB_pf5_uref100_infA_pf5_uref100_p1`:

   - Interactive plots in `iplot_*.html` HTML files viewable with any JavaScript-enabled browser.
   - Selected variants per criterion (see paper for details):
     - Text tables in `*.csv` files with data and metadata,
     - Pandas dataframes serialized and compressed in Binary tables compressed in `*.pkl.bz2` files.

   NOTE: This step will launch the previous app (`outparser`) as needed for all the CAMEOX runs belonging to a gene pair and frame. Depending on the number of runs and the variants per run, this may be time consuming. After that pre-processing step, it will process the resulting dataset containing results from all those runs. This dataset which may potentially range from hundreds to millions of entangled variants, with total processing time ranging from seconds to hours.

   More information on output data is given in the output section and in particular descriptions of the different criteria used to ranked solutions for experimental testing.

7.2.3 More advanced examples

Running parts of the pipeline interactively For debugging purposes, or for exploring more fine-grained options of each step from the tutorial above, it might be useful to start a shell within the CAMEOX singularity container image.

```
singularity shell --mount type=bind,source=/path/datangle, dst=/scif/data/cameox \
gentangle.sif
```

The bash commands ran at each step are located in `/scif/apps/*/scif/runscript`. You can run all the software tools used in this pipeline directly from the bash shell, such as `jackhmmer`, `hmmbuild`, `ccmpred`, `hhfilt`, `julia`, `python`, `jupyter` and many more. You can also do the same but initiating the shell for a specific app (point of entry to the container) with the `--app [appname]` to have a better environment for a specific program.

Using the analysis app of the container to run a customized downstream analysis notebook One can easily imagine a situation where a user may want to extend or modify the GENTANGLE downstream analysis Jupyter notebook, for instance, to add additional plots or to exclude some sampling methods. There is no need to any local installation of software with the next procedure:

1. Clone the GENTANGLE repository in a local directory, such as `/mypath/gentangle`
2. Copy `/mypath/gentangle/notebooks/Gentangle_downstream.ipynb` to a location that will be bind to the container, such as `/path/datangle/`:

   ```
   cp /mypath/gentangle/notebooks/Gentangle_downstream.ipynb /path/datangle/
   ```

3. Use your preferred editor to modify `/path/datangle/Gentangle_downstream.ipynb`, but please do not modify the name of the variables in the PARAMETERS cell, though you can modify their (default) value.
4. Open a containerized shell within the `analysis` app with:

   ```
   singularity shell --app analysis --mount type=bind,source=/path/datangle,\
   dst=/scif/data/cameox gentangle.sif
   ```

5. Process the modified notebook with `papermill` using the same parameters that you would pass to the `analysis` app, for instance:

   ```
   papermill --request-save-on-cell-execute --progress-bar -k python3 \
   /scif/data/cameox/Gentangle_downstream.ipynb /scif/data/cameox/output/ \
   /aroB_pf5_uref100_infA_pf5_uref100_p1/Gentangle_downstream_out.ipynb \
   -p PROTa aroB_pf5_uref100 -p PROT infA_pf5_uref100 -p FRAME p1 \
   -p POP_SIZE_THRESHOLD 500 -p ITERS_THRESHOLD 1 -p NUM_ERP 2 \
   -p SAMPLE_SIZE 250 -p OVERDEN_NUM 3 -p OVERDEN_PEN 0.2 -p OVERDEN_BIN 25
   ```

8 Funding

This work is supported by the U.S. Department of Energy, Office of Science, Office of Biological and Environmental Research, Lawrence Livermore National Laboratory BioSecure SFA within the Secure Biosystems Design program. Work at LLNL was performed under the auspices of the U.S. Department of Energy at Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.