CAMBer User's Manual

Release 1.0

Michal Wozniak

19 July 2011

Contents

1 About the software

1.1 Background

We observe a lot of inconsistencies in the genome structure annotations among bacterial strains (John Dunbar et al. BMC Genomics, 12(1):125, 2011). This inconsistency is a frustrating impedance to effective comparative genomic analysis of bacterial strains in promising applications such as gaining insights into bacterial drug resistance.

CAMBer (Wozniak M, Wong L, Tiuryn J. BMC Genomics 2011) is an approach to support comparative analysis of multiple bacterial strains. CAMBer unifies annotations of closely related species by homology transfer. It produces what we called multigene families. Each multigene family reveals genes that are in one-to-one correspondence in the bacterial strains, thereby permitting their annotations to be integrated. We present results of our method applied to three human pathogens: *Escherichia coli*, *Mycobacterium tuberculosis* and *Staphylococcus aureus*.

1.2 Availability

This software is an open source application (GPL 3 license). Sources are available at the code.google website:

`http://bioputer.mimuw.edu.pl/camber/software/camber2.zip`

Please don't hesitate to contact us with any comments and suggestion or if you are interested in co-developing this software.

1.3 About the authors

This software was implemented by Michal Wozniak. Project idea and guidance came from Limsoon Wong and Jerzy Tiuryn. Affiliation: Institute of Informatics, University of Warsaw

E-mails: `m.wozniak@mimuw.edu.pl`

2 Manual

2.1 Software requirements

- Python
- Sun Grid Engine (only for grid computations)

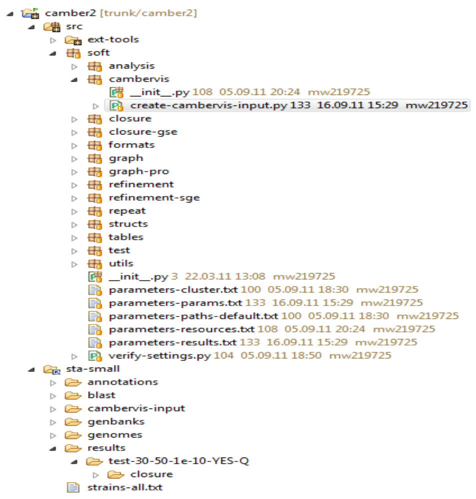
2.2 Installation

The CAMBer software is written in Python as a set of scripts. You need to have installed Python to run it. The software does not need classical installation, you only need to download and extract the zip package:

`http://bioputer.mimuw.edu.pl/camber`

2.3 Software design

The software is designed as a set of executable scripts written in Python which communicate each other via their outputs. Software's configuration assumes that an experiment data and results are stored on the same hierarchy level as the software. Figure 1 presents the hierarchy of both CAMBer and the *S. aureus* case study.



```
camber2 [trunk/camber2]
  src
    ext-tools
    soft
      analysis
      cambervis
        __init__.py 108  05.09.11 20:24  mw219725
        create-cambervis-input.py 133  16.09.11 15:29  mw219725
      closure
      closure-gse
      formats
      graph
      graph-pro
      refinement
      refinement-sge
      repeat
      structs
      tables
      test
      utils
      __init__.py 3  22.03.11 13:08  mw219725
      parameters-cluster.txt 100  05.09.11 18:30  mw219725
      parameters-params.txt 133  16.09.11 15:29  mw219725
      parameters-paths-default.txt 100  05.09.11 18:30  mw219725
      parameters-resources.txt 108  05.09.11 20:24  mw219725
      parameters-results.txt 133  16.09.11 15:29  mw219725
      verify-settings.py 104  05.09.11 18:50  mw219725
  sta-small
    annotations
    blast
    cambervis-input
    genbanks
    genomes
    results
      test-30-50-1e-10-YES-Q
        closure
    strains-all.txt
```

Figure 1: The folder hierarchy of the CAMBer software and the *S. aureus* case study.

Here we describe roles of the folders in Figure 1:

- `camber2` - CAMBer software
- `camber2/src/ext-tools` — BLAST (user's path to BLAST can be configured in `parameters-paths-defaults.txt`)
- `camber2/src/soft` — CAMBer source code
- `camber2/src/soft/cambervis` — contains a script to generate input data for CAMBerVis (`create-cambervis-input.py`)
- `camber2/src/soft/closure` — scripts to compute the closure procedure on the PC architecture
- `camber2/src/soft/closure-gse` — scripts to compute the closure on the SGE (Sun Grid Engine) architecture
- `camber2/src/soft/graph` — scripts to compute the consolidation graph before and after refinement
- `camber2/src/soft/formats` — a script to parse and convert GenBank annotations into simplifier annotations used by CAMBer
- `camber2/src/soft/refinement` — scripts to compute the refinement procedure on the PC architecture
- `camber2/src/soft/refinement-sge` — scripts to compute the refinement procedure on the SGE (Sun Grid Engine) architecture
- `camber2/src/soft/structs` — structures used by CAMBer
- `camber2/src/soft/utils` — implementation of common methods used by CAMBer
- `camber2/src/soft/parameters-params.txt` — parameters of the method to transfer annotations

- `camber2/src/soft/parameters-resources.txt` — parameters defining paths to store results
- `camber2/src/soft/parameters-paths-defaults.txt` — parameters of system dependent values like paths to the BLAST installation
- `sta` — folder with the *S. aureus* case study data
- `sta/annotations` — files with annotations
- `sta/genomes` — files with FASTA genome sequences
- `sta/genbanks` — files with GenBank genome annotations
- `sta/blast` — folder to store BLAST queries and results
- `sta/strains-all.txt` — list of *S. aureus* strains
- `sta/results` — CAMBer results
- `sta/results/all-30-50-1e-10-YES-Q` — CAMBer results for a specific set of parameters
- `sta/results/all-30-50-1e-10-YES-Q/closure` — CAMBer results for a specific set of parameters
- `sta/results/all-30-50-1e-10-YES-Q/ann` — annotations at all stages of the closure procedure
- `sta/results/all-30-50-1e-10-YES-Q/blast` — blast hits that are acceptable according to the parameters
- `sta/results/all-30-50-1e-10-YES-Q/graphs` — the consolidation graph, list of connected components, etc.
- `sta/results/all-30-50-1e-10-YES-Q/pseudo-ann` — gene annotations removed from consideration due to untypical start codon, stop codon or length.
- `sta/results/all-30-50-1e-10-YES-Q/refinement` — edges removed from the consolidation graph during the refinement procedure
- `sta/results/all-30-50-1e-10-YES-Q/iteration.txt` — current iteration of the closure procedure

2.4 Input files

Input files of CAMBer consist of genomes, annotations. There is also required one file defining a list of strains. According to the default parameters:

- annotations: stored in the `annotations` folder
- genomes: stored in the `genomes` folder
- list of strains: stored in the `strains-all.txt` file

2.5 Step 0: converting protein annotations from GenBank files

CAMBer uses simplifier annotations of genes encoding proteins — just keeps information about gene identifiers and gene locus. To retrieve such list of annotated protein coding genes run the GenBank file converting script:

`formats/genbanks_into_anns.py`.

2.6 Step 1: The closure procedure

Scripts to compute the closure procedure are stored in the folders: `camber2/src/soft/closure` (for the PC architecture) and `camber2/src/soft/closure-gse` (for the GSE architecture). In the first step to compute the closure procedure we prepare genome databases running: `prepare-databases.py`. Then, we prepare a list of annotations (filtering out these with untypical start codons, stop codons or lengths) running: `prepare-anns.py`. Next, we iteratively run the three scripts:

- `do-blasts.py` — compute BLASTs all strains against all
- `parse-blasts.py` — parse and filter BLASTs that do satisfy CAMBer parameters
- `merge-anns.py` — merge accepted BLAST hits with annotations after the last iteration

2.7 Step 2: The consolidation graph

The next step is to create the consolidation graph. To do this run the two following scripts:

- `graph/create-gene-graph.txt` - compute the graph of genes. It is an undirected graph. Nodes are predicted or annotated genes. There is an edge between a pair of genes if there is an acceptable BLAST hit between them.
- `graph/create-multigene-graph.txt` - compute the consolidation graph of multigenes. It is an undirected graph. Nodes are multigenes. There is an edge between two multigenes if there is an edge between at least two elements of these multigenes in the gene graph.

2.8 Step 3: Connected components - gene families

Before, we compute connected components there are required files to store details about multigenes. To compute it run this script:

- `graph/save-multigene-details.py`

Then, to compute connected components in the consolidation graph run the script: `graph/compute-components.py`.

To run the refinement procedure it is required to save details about the computed connected components, run: `genes/save-components-details.py`.

2.9 CAMBerVis: plug-in results into CAMBerVis

Once we have computed the first three steps we simply run the script:

- `cambervis/create-cambervis-input.py`

The script will generate the folder `sta/results/all-30-50-1e-10-YES-Q/cambervis-input`, which contains input files required by CAMBerVis (genome sequences, unified annotations and the list of strains). The simplest way to plug-in this data into CAMBerVis is to just copy the folder into `examples` forlder in the CAMBerVis installation directory.

2.10 Step 4: The refinement procedure (OPTIONAL)

Similarly to the closure procedure there are two versions of the scripts to compute the edges to remove in the refinement procedure, dependent to the platform: `refinement` for the PC platform and `refinement-sge` for the SGE (Sun Grind Engine).

The refinement procedure removes edges from the consolidation graph that correspond to pairs of paralogs rather than orthologs. For more details we refer to the publication about CAMBer.

To compute the set of edges to remove, run the script: `remove-edges.py`.

Again, run the sequence of scripts to build the consolidation graph and to compute connected components, after the refinement:

- `graph/create-multigene-graph-edges-removed.py`
- `compute-components-refined.py`
- `save-components-details.py`