***CoreCruncher***

**User Manual**

**Authors**

Connor D. Harris, Kasie Raymann and Louis-Marie Bobay

**Citation**

Please cite:

**Requirements and basic information**

Requires **Usearch** or **BLAST** (tested successfully with BLAST 2.7.1+)

Please download USEARCH: https://www.drive5.com/usearch/ or BLAST

################################################################

**IMPORTANT: Rename the usearch executable 'usearch61'**

**copy or move usearch61 into /usr/local/bin/**

**Or change the lines 176 to 181 in the script usearch\_core.py**

################################################################

OPTIONAL: can align core genomes and provide sequence concatenate if alignment program is available

Alignment program **muscle** or **mafft** must be executable and in /usr/local/bin

Requires the python library numpy

Simple example:

**python master.py -in example -out out\_folder**

**Required arguments:**

|  |  |  |
| --- | --- | --- |
| **-in** | PATH\_TO\_FOLDER | The input given to CoreCruncher is the path to a folder containing the genomes to analyze. |
| **-out** | OUTPUT\_FOLDER | CoreCruncher will create an output folder containg all the results of the analysis |

**Optional arguments**

|  |  |  |
| --- | --- | --- |
| **-freq** | Value  (default= 90) | Minimum frequency of the gene across genomes to be considered core (default= 90%, an ortholog is considered a core gene even if it is missing in 10% of the set of genomes)" |
| **-score** | Value  (default= 90) | Identity score used by usearch or blast to define orthologs in % (Default= 90%) |
| **-length** | Value  (default= 80) | -length Minimum sequence length conservation used by to define orthologs (default= 80%) |
| **-prog** | usearch or blast  (default= usearch) | Program to use to compare sequences: *Usearch* or *Blast* (default= usearch) |
| **-ref** | Genome file name  (default= first genome file in the folder) | Pivot genome (default: first genome in folder will be used as reference). If you want to specify the pivot genome to use, specify the name of the file in the folder (e.g. -ref genome1.prot) |
| **-id** | unique or combined  (default = combined) | Type of gene IDs in output files. Choose 'unique' if the same gene IDs are not found in different genomes or 'combined' to combine genome ID & gene ID (default= 'combined'). |
| **-ext** | .fa/.fasta/.prot/.faa or other  (default: will try to find it automatically) | File extensions of the genome files to analyze in the input folder; e.g. .fa/.fasta/.prot/.faa (default: will try to find it automatically) |
| **-list** | File name  (default= none) | Path to a file containing the list of genomes to analyze. This option can be used in case you don’t want to analyze all the genomes in your input folder (default: none, all the genomes in the folder will be analyzed by default) |
| **-restart** | yes or no  (default= no) | Restart analysis from scratch: yes or no (default= no). If yes is chosen, the program will erase the usearch output files and relaunch usearch or blast. Note that usearch or blast don’t need to be relaunched every time if you want to rebuild your core genome with different parameters. |
| **-align** | Program name  (default= none) | Align core gene sequences with specified program (muscle or mafft) and merge all the core genes into a single concatenate. Example= -align musclev0.0.0 or -align mafft)". Your alignment program must bin in your bin directory /usr/local/bin/ |
| **-stringent** | yes or no  (default= no) | Define a stringent core genome: yes or no (default= no). By default, core genes with paralogs will be conserved in the core genome and the paralogous sequences will be removed. If stringent is chosen, the core gene will be entirely removed from the core genome) |

**Ouput files**

All output files are written in the ouput folder at the location specified with the **-out** option.

**summary.txt** contains basic information about the inferred core genome and the parameters used to build the core genome.

**families\_core.txt** contains the list of the genes inferred as core. Each line corresponds to a core gene. The first column of the core gene indicates the name of the core gene given by CoreCruncher (e.g. fam1, fam2, etc…). The rest of each line corresponds to the gene ID of the sequence in the different genomes. By default, gene IDs are created by merging the genome name to the sequence ID separated by ‘&’, e.g. Serratia\_sp\_AS12.prot&WP\_000462905.1 indicates that sequence WP\_000462905.1 of genome Serratia\_sp\_AS12.prot. The IDs are combined in case the same gene ID is used across multiple genomes.

**core** is a folder containing all the sequences of each core gene in separate files. If the option **-align** is used, the folder also contains a aligned version of each core gene file with the extension **.align**.

**CC** is a folder containing all the files generated by *Usearch* or *Blast* (each genome compared to the pivot genome).

**The CoreCruncher process**

As opposed to most methods using Best Bidirectional Hits (BBH) CoreCruncher does not conduct all pairwise genome comparisons but robustly identifies core genomes based on the detection of “double outliers” to distinguish true orthologs from paralogs. In prokaryotes, the vast majority of paralogs are gained by HGT, and as a result, this introduces sequences that are expected to present atypical features relative to true orthologs. Our method first identifies homologous sequences by comparing each genome against a pivot genome. CoreCruncher uses *Usearch* (default) or *Blast* to identify homologs based on sequence identity and sequence length. For each gene sequence, each best hit is considered as a putative ortholog while other hits are directly classified as paralogs (i.e. in-paralogs). The putative orthologs are then assembled into orthologous gene families by transitivity (e.g. each pair of orthologs are automatically added to the same family). An orthologous family is considered a putative core gene only if there is a single gene copy per genome (when possible, paralogs are excluded, see below). These steps result in a putative core genome where no in-paralogs are present, but paralogs might still be included due to more complex patterns of gene gains and losses or incomplete genome assemblies resulting in seeming orthologs (i.e. “hidden paralogs”). These cases are expected to happen when the orthologous sequence is lost but a paralog remains present in the genome and current methods based on BBH are unlikely to recognize these sequences as paralogs.

To ensure that no hidden paralogs are included in the core genome, our method identifies sequences that are significantly more divergent from the other sequences of the orthologous gene, while accounting for the overall divergence of each genome. A given gene sequence can present a higher divergence rate relative to the other sequences of the orthologous gene, but this can simply be due to the fact that this gene sequence is from a more divergent strain. To account for this, we exclude an orthologous gene from the core genome if it contains a double outlier: a sequence that is substantially more divergent from i) the other sequences of the orthologous gene family and ii) more divergent than the other putative orthologs of the genome.

First, for each putative core gene, the distribution of identity scores of all sequences (Figure 2: distribution 1) is built and compared to the identity scores of the in-paralogs, i.e. each sequence is considered as an ortholog unless an in-paralog with a higher identity score has been identified. In the case where an ortholog presents one or more sequences with lower identity score than an in-paralog, the low-identity sequences are excluded from the ortholog family, which will still be considered a putative core gene if it meets the frequency criterion (i.e. by default an ortholog must be present in 90% of the genomes to be considered a core gene). When CoreCruncher is run with the stringent option, an ortholog is automatically excluded from the core genome if a sequence with a lower identity score than an in-paralog is detected. Obviously, this first step is only conducted when in-paralogs have been identified.

Second, for each genome compared to the pivot genome, the set of putative core gene sequences of this genome is used to draw the distributions of identity scores (distribution 2). The median value of each distribution is used to estimate the overall divergence between each genome and the pivot genome. Then, for each sequence of each putative core gene, we test for the presence of “double outliers”. A double outlier is defined as a sequence that is significantly divergent i) *vertically*: from the other sequences of the orthologous gene (using distribution 1) and ii) *horizontally*: from the average identity score computed across all the putative orthologs relative to the pivot genome (using distribution 2). In both cases, a sequence is defined as an outlier with Tuckey’s fences {Tukey, 1977 #2508}: if its identity threshold is below Q1-1.5(Q3-Q1) or above Q3+1.5(Q3-Q1), with Q1 and Q3 the values or of the first and third quartiles, respectively. When a given sequence is inferred as a double outlier, it is excluded from the putative core genome, which will still be considered part of the final core genome if it still meets the frequency criterion (i.e. by default an ortholog must be present in 90% of the genomes to be considered a core gene). When CoreCruncher is run with the stringent option, a putative core gene is automatically excluded from the core genome is it contains one or more sequences inferred as a double outlier. After filtering out putative core genes with the double outlier test, the final core genome is built.

**Recommendations**

*CoreCruncher* does not compute all pairwise comparisons of genomes, which makes it very efficient. Instead *CoreCruncher* uses a pivot genome against all other genomes are compared. We highly recommend using a good quality genome as the pivot genome, since a low-quality pivot genome can result in a substantially smaller core genome. The file name of the genome to use as pivot can be specified with the **-ref** option. The pivot genome must be in the input folder with the other genomes.