***CoreCruncher***

**User Manual**

**Authors**

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

**Citation**

Harris et al. MBE 2020.

**Contact**

ljbobay [at] uncg [dot] edu

**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 corecruncher\_master.py -in input\_folder -out out\_folder -ext .fa**

Note that you don’t need to specify the extension of the genome files if a single type of files is present in the input folder but we strongly recommend using this option.

**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) |
| **-batches** | Integer  (default = 1) | Split the dataset into multiple batches of genomes. When building large core genomes on thousands of genomes, computers can run out of memory. To avoid this issue, this option allows to divide the dataset into multiple batches of genomes and the core genome is built for each batch independently. Each batch is built using the same pivot genome. *CoreCruncher* then builds the overall core genome from the different core genomes. Our experience suggests that a desk computer can typically handle 1,000 to 5,000 genomes at once (depending on the computer). |

**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).

**double\_outliers.txt** contains all the gene families trimmed or excluded from the core genome based on the double outlier test. Note than when *CoreCruncher* is run with the non-stringent option, the inferred paralogs will be excluded from some of these gene families and the trimmed family will be part of the core genome (if it is still large enough to be part of the core genome).

**The CoreCruncher process**

Due to the accumulation of sequencing data, it has become common place to analyze hundreds to thousands of complete genome sequences during the study of a single prokaryotic species (Parks, et al., 2018). Most algorithms implemented to define orthologous genes first rely on pairwise genome comparisons; a task that is becoming increasingly difficult to complete as datasets grow in size (Kristensen, et al., 2011). To circumvent this issue, we have developed an approach that does not conduct all pairwise genome comparisons and instead, robustly identifies core genomes based on our “double outliers” approach to distinguish true orthologs from paralogs.

In prokaryotes, the vast majority of paralogs are gained by horizontal gene transfer (HGT) (Treangen and Rocha, 2011) which leads to the introduction of sequences that are expected to present atypical features relative to true orthologs (i.e. typically more divergent sequences). To robustly exclude paralogs, our method first identifies homologous sequences by comparing each genome against a pivot genome. Specifically, *CoreCruncher* uses *Usearch* (Edgar, 2010) (default) or *Blast* (Altschul, et al., 1997) to identify homologs based on sequence identity and sequence length. For each gene sequence of the pivot genome, each best hit is considered as a putative ortholog while other hits are directly classified as paralogs (i.e. within-paralogs). All of the best hits to each gene of the pivot genome constitute a putative ortholog and, as such, are associated together in an orthologous family. The orthologous family is considered a putative core gene when found in all or nearly all genomes (90% of the genomes by default). This step ultimately results in a putative core genome where no within-paralogs are present however; paralogs may 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 occur when the orthologous sequence is lost, but a paralog remains present in the genome (Figure 1). Current methodologies, based on BBH, are unlikely to recognize these sequences as paralogs and may include them in the core genome (Kristensen, et al., 2011).

Firstly, *CoreCruncher* identifies partially hidden paralogs as illustrated in Figure 1C**.** Paralogous genes can be hidden paralogs in some genomes (in instances where the orthologous sequence is absent from the genome) and within-paralogs in other genomes. These cases are relatively straightforward to identify: For each putative core gene, the distribution of identity scores of all sequences is built and compared to the identity scores of the within-paralogs, i.e. each sequence is considered to be an ortholog unless an within-paralog with a higher identity score has been identified. In the case where an ortholog presents one or more sequences with a lower identity score than an within-paralog, the low-identity sequences are excluded from the orthologous family - which will still be considered a putative core gene if it meets the frequency criterion (i.e. by default an orthologous family must be present in 90% of the genomes to be considered a putative core gene). When *CoreCruncher* is run with the *stringent* option, an orthologous family is automatically excluded from the core genome if a sequence with a lower identity score than an within-paralog is detected. Note that this step is only conducted when within-paralogs have been identified for a given orthologous gene family.

Secondly, *CoreCruncher* identifies completely hidden paralogs as represented in Figure 1D. Completely hidden paralogs are hidden paralogs in one or more genomes (orthologous sequence is absent from the genome(s)) without any within-paralogs present in other genomes. 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 may present a higher divergence rate relative to other sequences of the orthologous gene, but this may simply be due to the fact that this gene sequence is present in a more divergent strain. To account for this, we exclude sequences, or an orthologous family, from the core genome if it is itself, or if it contains, a “double outlier”, which is defined in this study as a sequence that is substantially more divergent from i) the other sequences of the orthologous gene family (Figure 2, distribution 1) and ii) more divergent than the other putative orthologs of the genome (Figure 2, distribution 2). The set of putative core gene sequences is used to draw the distributions of identity scores for each genome that is compared to the pivot genome (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”, which, as defined above, is 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): if its identity threshold is below Q1-1.5(Q3-Q1) or above Q3+1.5(Q3-Q1), with Q1 and Q3 the values of the first and third quartiles, respectively. When a given sequence is inferred as a double outlier, it is considered a hidden paralog and this genome’s sequence is excluded from the putative core gene. Other sequences of the putative core gene will still be considered part of the final core genome if they meet 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 if it contains one or more sequences inferred as a “double outlier”. After filtering out paralogous sequences and/or 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.

As mentioned above the choice of the pivot genome can affect the core genome built by *CoreCruncher*, especially when these genomes are not well assembled and/or when some genes are missing from the pivot genome. To compensate for the use of poor-quality pivot genomes, *CoreCruncher* comes with a utility script *consensus.py*. This script allows to make the consensus core genome from two core genomes built with different pivot genomes. The script checks for potential inconsistencies in the two core genomes and removes inconsistent core genes. Potential core genes missing in one of the two core genomes due to the absence of a sequence in one or the pivot genomes are re-introduced in the consensus core genome. Note that the two core genomes must be built with *CoreCruncher* using the exact same parameters but with different pivot genomes. Using this procedure usually yileds slightly larger core genomes. The consensus script can then we used as follow:

**Python consensus.py -in genome\_folder -folders FOLDER1 FOLDER2 -out outpout\_folder**

With:

**genome\_folder** the folder containing the genomes originally given to *CoreCruncher*

**FOLDER1** and **FOLDER2** the two output folders of *CoreCruncher* (each built with a different pivot genmoe)

**outpout\_folder** the outpout\_folder where the consensus core genome will be written along with the sequences of each core gene.

*CoreCruncher* yields conservative core genomes compared to BBH-based approaches. Because all the comparisons are made against a single pivot genome, most core genes defined by *CoreCruncher* present a narrower range of identity scores than BBH-based approaches. We recommend using more relaxed identity thresholds than those typically used for BBH-based approaches, e.g. 90% instead of 95%. Our tests suggest that using lower identity thresholds does not negatively impact the quality of the core genome, but this parameter is rather species or clade-specific. Datasets comprising more divergent genomes should be built using more relaxed identity scores.