Zebra3D Parameters

The complete list of Zebra3D parameters is provided below. Three parameters are mandatory and specify the input data and output storage. The algorithm itself runs on default settings which were either inherited from the ideologically close sequence-based **Zebra/Zebra2** SSPs/SDPs-prediction method, or derived automatically using well-established general-purpose procedures. All numerical parameters can be set/adjusted by the user for a particular purpose.

List of parameters:

Mandatory input parameters:
- **aligned_pdbs**
- **aligned_fasta**
- **output**

Utilization of computing resources:
- **cpu_threads**

Cluster analysis methods:
- **method**
- **eps**
- **min_size_of_subfamily**

Selection of common core positions:
- **max_content_of_gaps**
- **max_content_of_mismatch**
- **mismatch_threshold**

Filtering/postprocessing parameters:
- **ref**
- **max_ssr_length**
- **max_outliers**
- **exclude_ncterm**

Special case options:
- **ssr_start**
- **ssr_end**

PyMol parameters:
- **compile_pymol_pse**

[return to full list]

aligned_pdbs

Group: Mandatory input parameter
Value type: `string`
Default: *`null`*
Example: `aligned_pdbs=./aligned_pdbs`
Comment: Path to folder with aligned protein 3D-structures as separate files in the PDB format (each file should represent one chain). Preparation of the input data is discussed **here**.

[return to full list]

aligned_fasta

Group: Mandatory input parameter
Value type: `string`
Default: *`null`*
Example: `aligned_fasta=./alignment.fasta`
Comment: Path to the corresponding sequence representation of the alignment in the FASTA format. Preparation of the input data is discussed **here**.

[return to full list]

output

Group: Mandatory input parameter
Value type: `string`
Default: *`null`*
Example: `output=./results`
Comment: Path to folder to store results. The folder must be empty or null (i.e. it should not exist at the time of command execution). In the latter case, the new folder will be automatically created by the program. In case the folder already exists and is not empty, the program will terminate with an error to avoid overwriting previous results, as this could cause a mess in your data.

[return to full list]

cpu_threads

Group: Utilization of computing resources
Value type: `int`
Default: *`all`*
Example: `cpu_threads=10`
Comment: Number of parallel CPU threads to utilize. By default, all physically available CPU threads will be automatically detected and utilized. As Zebra3D analysis is resource consuming and may take some time, utilization of 100% CPU resources may slow down all other processes in your system. E.g., if you run the program on your desktop station with the default setting, you may not be able to use other software (surf the internet, edit documents, etc.) until Zebra3D calculations complete. To avoid such inconvenience, you may allocate only a fraction of resources to Zebra3D. E.g., if your system contains 8 physical cores and 16 threads, you may give 10 to Zebra3D and leave 6 to yourself.

[return to full list]

method

Group: Cluster analysis methods
Value type: `[hdbscan|optics|dbscan]`
Default: `hdbscan`
Example: `method=optics`
Comment: Select the cluster analysis method. By default, the fully automated HDBSCAN algorithm is used to produce "thicker" clusters and minimize the amount of outliers, thus preserving as much data as possible for further expert analysis. Two alternative methods can be switched on for a particular purpose. The OPTICS is a fully automatic technique that tends to produce more spatially consistent (compact) "thinner" clusters at the cost of data loss by throwing out a larger number of proteins as outliers. Then, the DBSCAN is a curated technique dependent on the ‘eps’ parameter that can be manually calibrated to meet the particular research objective (see below).

[return to full list]

eps

Group: Cluster analysis methods
Value type: `float`
Default: *`null`*
Example: `eps=1`
Comment: Set the eps>0 to fine-tune the output of DBSCAN cluster analysis algorithm.

[return to full list]

min_size_of_subfamily

Group: Cluster analysis methods

Value type: `int` (number of proteins)
Default: *`automatically set to 10% of the total number of proteins`*
Example: `min_size_of_subfamily=3`
Comment: The purpose of this parameter is to regulate the minimal size of a subfamily/cluster within SSR. The exact implementation of this parameter depends on the choice of the machine-learning cluster analysis method. When HDBSCAN is used, the value assigned to this Zebra3D's parameter is passed to the `min_cluster_size` option of that algorithm. The HDBSCAN parameters are explained **here**. When OPTICS or DBSCAN are used, the value is passed to the `min_sample` option, which regulates, for each SSR, the number of fragments of local 3D-structure in a neighborhood of the current fragment for it to be considered as the core point for clustering (the current fragment included). The OPTICS and DBSCAN parameters are explained **here** and **here**, respectively. By default, this parameter is set to 10% of the total number of PDB entries in the input alignment, but not less than 2 proteins. Selection of the default value for this parameter was based on the equivalent parameter in the ideologically close **Zebra2 tool** (i.e. "Min size of a subfamily").

[return to full list]

max_content_of_gaps

Group: Selection of common core positions
Value type: `int`, %
Default: `5`
Example: `max_content_of_gaps=5`
Comment: Define the allowed gap content in alignment column, in % from the total number of the superimposed proteins, for it to be considered as a "common core" position.

[return to full list]

max_content_of_mismatch

Group: Selection of common core positions
Value type: `int`, %
Default: `5`
Example: `max_content_of_mismatch=5`
Comment: Define the allowed 3D-mismatch content in alignment column, in % from the total number of the superimposed proteins, for it to be considered as a "common core" position. By default, the threshold to discriminate spatially aligned from misaligned residues will be selected automatically using the "elbow method" heuristic - i.e. as the bending point indicating the most significant change in the ascending trend of the RMSD metric between fragments within the SSR. See detailed explanation of the procedure in the **Zebra3D publication**. Alternatively, a hard cut-off value can be set by the user (see below).

[return to full list]

mismatch_threshold

Group: Selection of common core positions
Value type: `float`, angstroms
Default: *`null`*
Example: `mismatch_threshold=5`
Comment: Define the cut-off value to discriminate spatially aligned from misaligned residues, in angstroms. By default, the threshold is selected automatically, specific to the input alignment (see above).

[return to full list]

ref

Group: Filtering/postprocessing parameters
Value type: `string`
Default: *`first`*
Example: `ref=0_1bvt_A`
Comment: Set the reference protein by its name in the FASTA alignment. By default, the first protein in the FASTA alignment file will be selected as the reference. The numbering of amino acid residues in the reference protein will be used in various output files to specify the location of SSRs; its structure will be used to prepare one of the 3D-annotation files, containing the summary map of all identified SSRs. Otherwise, selection of the reference protein has not effect on calculation/evaluation/selection of SSRs.

[return to full list]

max_ssr_length

Group: Filtering/postprocessing parameters
Value type: `int`
Default: `9999`
Example: `max_ssr_length=20`
Comment: Maximum number of positions (residues) in a region. SSRs of larger size will be dismissed. The size of an SSR is calculated as the maximum value of average length of protein fragments within each cluster/subfamily. This filter is disabled by default, i.e. set to 9999.

[return to full list]

max_outliers

Group: Filtering/postprocessing parameters
Value type: `int`, %
Default: `100`
Example: `max_outliers=40`
Comment: Maximum number of outliers, in % from the total number of protein in the input alignment. SSR with a larger % of outliers will be dismissed. This filter is disabled by default, i.e. set to 100%.

[return to full list]

exclude_ncterm

Group: Filtering/postprocessing parameters
Value type: `int`
Default: *`null`*
Example: `exclude_ncterm=5`
Comment: Dismiss SSRs that assign N-/C-terminal regions (first and last N residues of any PDB entry) to subfamilies. These parts of PDB entries may be inaccurate due to limitations of the experimental methods and may contain very mobile residues not related to a function. Thus, turning this filter on may improve ranking of true functionally important SSRs. E.g., if set to `5`, SSRs containing the first five and the last five residues of any PDB entry included into the proposed subfamily classification will be dismissed (i.e., not ranked in the output list). It is important to understand, that this is a *post*processing filter that is applied only to proteins included into the subfamily classification; in particular, an SSR will NOT be dismissed, if the N-/C-terminal fragment belongs to an outlier. By default, N-/C-terminal regions are included in the analysis to preserve as much data as possible for further expert assessment.

[return to full list]

ssr_start

Group: Special case options
Value type: `int`
Default: *`null`*
Example: `ssr_start=50`
Comment: Evaluate only one selected region as potential SSR by specifying the IDs of first (this parameter) and last (next parameter) residue as in the reference protein.

[return to full list]

ssr_end

Group: Special case options
Value type: `int`
Default: *`null`*
Example: `ssr_end=60`
Comment: Evaluate only one selected region as potential SSR by specifying the IDs of first (previous parameter) and last (this parameter) residue as in the reference protein.

[return to full list]

compile_pymol_pse

Group: PyMol parameters
Value type: `[true|false]`
Default: `true`
Example: `compile_pymol_pse=false`
Comment: Disable compilation of PyMol sessions with 3D-annotation of results (enabled, by default). If your input alignment is very large, compilation of all PyMol 3D-annotations can take considerable time. With this parameter set to `false`, only the PyMol instruction '.py' files will be generated in no time.

Then, only the selected 3D-annotations can be compiled by manually running PyMol using the respective instruction '.py' file: `pymol -qc RESULTS.py` or `pymol -qc ssr_rank.py`.