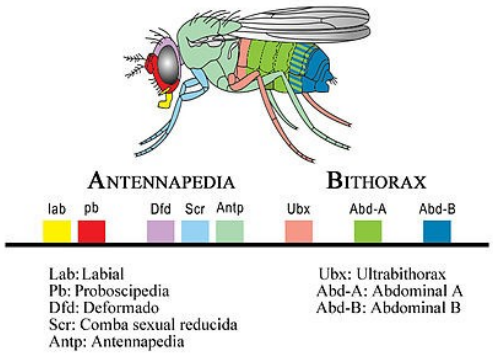
ClusterScan 0.1.0 Manual

ClusterScan is a tool which search for clusters starting from a feature annotation. It allow the user to scan an annotation file (BED format) and get clusters coordinates in output. ClusterScan also need an additional two-columns file storing the feature names and the corresponding accession for the cluster search including Gene Ontology, KEGG, Pfam accessions, etc. The user can also use a custom accession, making ClusterScan very flexible.

Taking genes, for example, a gene cluster is a set of two or more genes closely located on the same chromosome and that encode for similar proteins sharing a generalized function. An example of a gene cluster is the Hox gene, which is composed of eight genes in *Drosophila melanogaster* and is part of the Homeobox gene family:



The size of gene clusters can vary significantly, from a few genes to several hundred genes. Since the mission of ClusterScan is to allow users to search for any kind of features which are organized in clusters, it accept any kind of annotation and accession (custom made included).

How the tool works:
ClusterScan is composed by two different algorithms that perform the search but it also offer filters to select the minimum number of features to validate a cluster (that can't be a number lower than 2). The two algorithms are:

- clusterdist: scans the features using bedtools merge in order to find those features which are separated by a maximum distance in base pairs that can be selected by the user (p: --distance). Some studies based on gene families, for example in human and mouse, have estimated genes within 500 kb to be in cluster (Niimura et al. 2003; Tadepally et al. 2008).

- clustermean: divides the genome in sliding windows and calculates the mean number of features and the standard deviation for each accession. After that, clustermean searches for those windows containing a number of features higher or equal to the relation *mean+n*stdv* in which *n* can be set by the user (p: --seed). Thus the *seed* parameter set the number of standard deviations to identify a window which serves as the beginning of the cluster. After this step, the algorithm similarly tries to extend the cluster in both the direction starting from the seed using the same relation *mean+n*stdv* in which *n* can be set by the user (p: --extension). Thus the *extension* parameter set the number of standard deviations to identify the window(s) which serve to extend the cluster. Lastly, clustermean trims the clusters in order to replace the cluster start/end represented by the first window start and the last window end respectively with the first gene in cluster start and the last gene in cluster end.

1. Niimura Y, Nei M: Evolution of olfactory receptor genes in the human genome. Proc Natl Acad Sci USA 2003, 100(21):12235-40.
2. Tadepally, H. D., Burger, G. & Aubry, M. Evolution of C2H2-zinc finger genes and subfamilies in mammals: Species-specific duplication and loss of clusters, genes and effector domains. BMC Evol. Biol. 8, 176 (2008).

Dependencies:
ClusterScan requires Python (v.2.7.x). Bedtools (v.2.25.0+) and R (v.3.0.0+) are needed to be in the user path. It also need some other Python libraries which can be easily installed via pip (https://pip.pypa.io/en/stable/installing/) from PyPI (https://pypi.python.org/pypi):

pybedtools
pandas
rpy2

Finally, in order to draw high quality clusters distributions for features in the top 10 clusters found (by number of features), it is also required to install the R library ggplot2 (v.2.0.0+).

Options:
ClusterScan can be launched specifying the name of the algorithm to be ran (clusterdist or clustermean) as positional argument. The way the two algorithms perform the search can be configured using several specific options but there are also some shared options.

Clusterdist:

-d, --dist=<bp>	Maximum distance between features in bp [default: 500000].

clustermean:

-w, --window=<bp>	Window size [default: 500000].
-s, --slide=<bp>	Sliding size [default: 250000].
-k, --seed=<n>	Number of standard deviations to identify a window which serves as the beginning of the cluster [default: 3].
-e, --extension=<n>	Number of standard deviations to identify the window(s) which serve to extend the cluster [default: 2].

shared options:

-o, --output PATH	Specify output path [default: ./].
-a, --analysis NAME	Specify optional analysis name for output files.
-n, --nf=<n>	Minimum number of features per cluster [default: 2].
--info FILE	Specify optional file to describe accessions.

An execution example:
clusterscan.py clusterdist my_genes.bed my_accessions.txt -d 250000 -a Analysis_01

I/O:
ClusterScan is composed by two files: *clusterscan.py* and *docopt.py*. The first represent the tool itself while the latter is a dependence that provides an interface for the command-line menu. The two files need to be in the same folder.

ClusterScan requires two mandatory files and an optional file in input. The first mandatory file is represented by a six-field bed file of feature annotations. It may follow the classical organization of the BED file format (https://genome.ucsc.edu/FAQ/FAQformat.html#format1). The second file is a two-column table containing the name of the features on the first column and an accession for which the cluster search need to be called. This accession can belong to Gene Ontology, KEGG, Pfam, etc. The user can also search for clusters based on a custom made series of accessions. It is not uncommon that a feature is associated with different accessions at the same time and in this case the feature can take part of multiple different kind of clusters.

ClusterScan gives six files in output:

**clusters.csv* stores the coordinates of all the clusters found. It contains an ID for each cluster; the accession (ACC) for which the cluster was composed; the chromosome/scaffold (chr) on which it resides; its start and end coordinates; the number of features within the cluster (n_features) and the number of features which overlap the cluster but belong to a different kind of accession (n_bystanders).

**clusters.bed* a bed version of the previous file for bedtools/bedops compatibility. Fields respect the format and are (from first to last column): chromosome/scaffold on which the cluster resides; its start/end coordinates using a 0-based start and 1-based end system of coordinates; the cluster ID; the number of features within the cluster; the strand; the cluster accession.

**summary.csv* contains a per-accession summary of the cluster analysis with the accession (ACC); the number of clusters found for that accession (n_clusters) the total number of features found for clusters belonging to that accession (n_ft); the number of bystanders (n_bs); the number of features and bystanders found in the clusters which contains the minimal and the maximal number of these features (max_ft and min_ft / max_bs and min_bs).

**features.csv* is a list of features found to be in overlap with clusters. It contains the chromosome/scaffold (chr) on which it resides; its start and end coordinates; the feature name; the score; the strand; an ID of the cluster on which it overlaps (cluster_ID); the accession of the cluster in which it overlaps (cluster_ACC).

**bystanders.csv* is a list of bystanders found to be in overlap with clusters. It contains exactly the same fields described for the file *features.csv*.

**distribution.pdf* is an histogram which shows the distribution of features in the per-accession top-10 clusters by number of features.

Differences between ID and ACC:
In ClusterScan the word “ID” refers to the cluster identifier. An ID for each cluster is assigned during the analysis joining the “C” letter with a progressive number starting from 1. Contrariwise the word “ACC” refers to the database accessions used to classify the features. They can come from Gene Ontology, KEGG, Pfam, etc. but the user can also use its own custom accessions and describe them using a third two-column tab-delimited txt file (ACC, description) that can be read by the program through the --info parameter (see the tutorial below and the *Pfam-descriptors.txt* file).

Tutorial:
in the tutorial folder you can find a dataset useful to run a cluster search with ClusterScan. There are three files in that folder:

Tetraodon_nigroviridis_8.88_genes.bed all genes from Tetraodon are annotated and stored in this bed formatted file. The bed file was obtaining parsing the gff3 annotation file (assembly version: 8.0; ensembl version 88) looking for protein coding genes (type: “gene” at column 3; biotype: “protein_coding” at column 9). Source can be downloaded at ftp://ftp.ensembl.org/pub/release-88/gff3/tetraodon_nigroviridis/Tetraodon_nigroviridis.TETRAODON8.88.gff3.gz

Tetraodon_nigroviridis_8.88_Pfam.txt Pfam domain annotations are stored for each protein coding gene in a two-column tab-delimited table. The file was obtained through BioMart (www.ensembl.org/biomart) selecting:
Database: Ensembl Genes 88
Dataset: Tetraodon genes (TETRAODON 8.0)
Filters: Gene type: protein_coding
Attributes: Gene stable ID
Pfam domain ID

Pfam-descriptors.txt all Pfam accession descriptions are stored in this two-column tab-delimited table. It can be used with the --info parameter in order to add a brief description of the cluster ACC in the **summary.csv* output.

You can run your first ClusterScan analysis and check that the program is working properly by typing:

```
python clusterscan.py clusterdist ./tutorial/Tetraodon_nigroviridis_8.88_genes.bed
./tutorial/Tetraodon_nigroviridis_8.88_Pfam.txt -d 86647 -a Tetraodon_01
```

in your terminal, from the folder in which you have extracted ClusterScan. Here is also a folder named “results” in which they are stored output files for exactly the same analysis.