MetaCHIP User Manual

Centre for Marine Bio-Innovation, University of New South Wales

November 1st, 2018

songwz03@gmail.com

Introduction

MetaCHIP is implemented in Python, a list of dependencies needs to be installed before running. Details of these dependencies can be found at: https://github.com/songweizhi/MetaCHIP.

To install MetaCHIP simply download the package and run the programs from the command line interface. **Full path to a list of dependencies needs to be specified in the config.txt file if they are not in environment variables**; otherwise, keep the config.txt file as it is.

MetaCHIP's input is the sequence file of a set of genome bins derived from metagenomic data. Please make sure the length of the sequence ID in all input genome bins is **NO LONGER THAN 22 letters.**

The MetaCHIP pipeline contains three scripts: Get_clusters.py, Best-match.py and Phylogenetic.py.

1. **Get_clusters.py** clusters input genome bins into sub-groups according to their phylogenetic relationships.
2. **Best-match.py** performs the best-match approach.
3. **Phylogenetic.py** performs the phylogenetic approach.

Get_clusters.py

```
-i          input genome folder
-x          file extension
-p          output prefix
-dc         distance cutoff
-taxon      taxonomy classification of input genomes
-tr         taxon ranks, e.g. d (domain), p (phylum), c (class),
            o (order), f (family)
-tuning     specify to run clustering with provided cutoff, while
            skipping previous steps
```

Get_clusters.py will cluster input genomes into sub-groups based on a phylogenetic tree derived from protein sequences of 43 single-copy genes (SCG) from CheckM [1].

Clustering profile generated in this step **should be manually curated** by comparing it with the SCG tree or taxonomic classifications of the input genomes (if available) prior to the HGT identification step. You can do this by changing the group assignment of input genome bins specified in the first column of [prefix]_grouping_g[num].txt file. Or, you can modify clustering sensitivity by re-run this step with a customized `distance cutoff` (`-dc`) after had a look at the [prefix]_grouping_g[num].png file. The "`-tuning`" option need to be specified to skip gene prediction and tree building steps. An example of the taxon classification file can be found at folder "`example_dataset`" together with the scripts.

Example command:

```
# default setting
$ python Get_clusters.py -i human_gut_bins -x fna -p human_gut

# with modified distance cutoff and taxon information
$ python Get_clusters.py -i human_gut_bins -x fna -p human_gut -dc 1.2 -tuning -taxon taxon_classification.tsv -tr c
```

Output files:

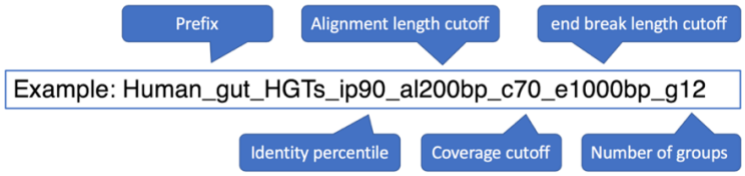
1. Clustering results were exported to [prefix]_grouping_g[num].txt.
2. SCG trees ([prefix]_grouping_g[num]_tree.jpg) of the input genome bins.
3. A dendrogram ([prefix]_grouping_g[num].png) showing the hierarchical clustering of input genome bins.

The **[prefix]_grouping_g[num].txt** file will be used as input for running Best-match.py and Phylogenetic.py.

Best-match.py

```
-p                output prefix
-g                grouping file
-blastall         all vs all blast results
-cov              coverage cutoff
-al               alignment length cutoff
-flk              the length of flanking sequences to plot
-ip               identity percentile cutoff
-eb               minimal length to be considered as end break
-tmp              keep temporary files
-num_threads      number of threads for running blastn
```

HGT candidates predicted by the best-match approach, as well as the plots of their flanking regions are exported to a folder with name in the following format:



Example command:

```
# default setting
$ python Best-match.py -p human_gut -num_threads 9
```

Output files:

A list of HGT candidates identified by best-match approach are exported to **HGT_candidates_BM.txt**. Their nucleotide and amino acid sequences are exported to HGT_candidates_BM_nc.fasta and HGT_candidates_BM_aa.fasta.

Phylogenetic.py

```
-p          output prefix
-g          grouping file
-cov        coverage cutoff
-al         alignment length cutoff
-ip         identity percentile
-eb         the minimal length to be considered as end break
-a          Prokka output
-o          orthologs folder
```

All protein orthologs within the input genomes need to be obtained for the phylogenetic approach, you can get it with GET_HOMOLOGUES [2]. The input is a bunch of annotation files for input genome bins in Genbank format, which have been generated by Get_clusters.py (**[prefix]_gbk_files**).

Example command:

```
# run GET_HOMOLOGUES
$ cd human_gut_MetaCHIP_wd
$ get_homologues.pl -f 70 -t 3 -S 70 -E 1e-05 -C 70 -G -n 16 -d human_gut_gbk_files

# default setting
$ python Phylogenetic.py -p human_gut
```

Output files:

HGT candidates validated by phylogenetic approach are exported to the same folder as best-match approach.

1. **HGT_candidates_PG.txt**: Best-match approach predicted HGTs, with additional information provided by phylogenetic approach.
2. **HGT_candidates_PG_validated.txt**: Phylogenetic approach validated HGTs only.
3. **HGT_candidates_PG_aa.fasta**: Nucleotide sequences of phylogenetic approach validated HGTs.
4. **HGT_candidates_PG_nc.fasta**: Amino acid sequences of phylogenetic approach validated HGTs.

References

1. Parks DH, Imelfort M, Skennerton CT, Hugenholtz P, Tyson GW: CheckM: assessing the quality of microbial genomes recovered from isolates, single cells, and metagenomes. Genome research 2015, 25:1043-1055.
2. Contreras-Moreira B, Vinuesa P: GET_HOMOLOGUES, a versatile software package for scalable and robust microbial pangenome analysis. Applied & Environmental Microbiology 2013, 79:7696-7701.