MetaCHIP User Manual

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

November 10th, 2018

songwz03@gmail.com

1. Introduction

MetaCHIP is implemented in Python and a list of dependencies needs to be installed (https://github.com/songweizhi/MetaCHIP).

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

The input files for MetaCHIP include a folder that holds the sequence file (in FASTA format) of all query genomes, as well as a text file, which holds taxonomic classification of all input genomes. Please make sure the length of sequence IDs for all input genomes is **NO LONGER THAN 22 letters**.

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

1. **PrepIn.py** - prepare input files
   1.1 Grouping input genomes according to their taxonomic classifications and user provided taxonomic rank.
   1.2 Gene calling from the input genomes with Prodigal.
   1.3 Build phylogenetic tree for input genomes according to the protein sequences of 43 single-copy genes (SCG) from CheckM [1].

2. **Best-match.py**
   Performs the best-match (BM) HGT identification.

3. **Phylogenetic.py**
   Performs the phylogenetic (PG) HGT identification.

2. PrepIn.py

```
PrepIn.py -h
-i            input genome folder
-t            taxonomic classification results
-l            taxonomic ranks for grouping, options including k(Kingdom), p
              (Phylum), c(Class), o(Order), f(Family) and g(genus). Default is c.
-x            sequence file extension, default is fasta
-p            output prefix
-threads      number of threads, default is 1
```

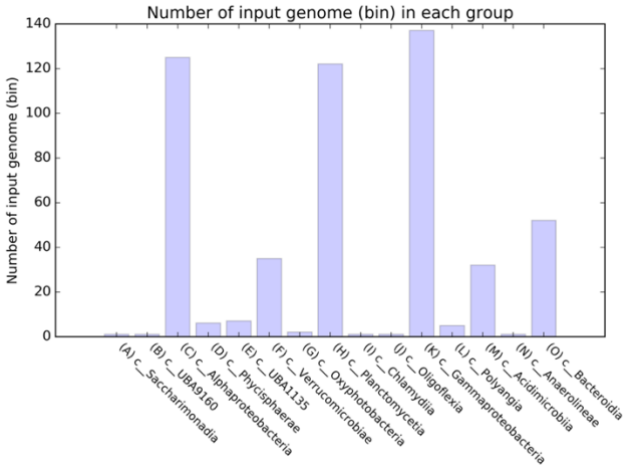
Get_clusters.py will group input genomes at defined taxonomic rank according to their taxonomic classification results. GTDBTk (https://github.com/Ecogenomics/GTDBTk) is recommended for taxonomic classification of input genomes. An example of the taxonomic classification file can be found at folder "`example_dataset`".

Example command:

```
$ python PrepIn.py -i human_gut_bin -t human_gut.summary.tsv -l c -x fna -p human_gut
```

Output files:

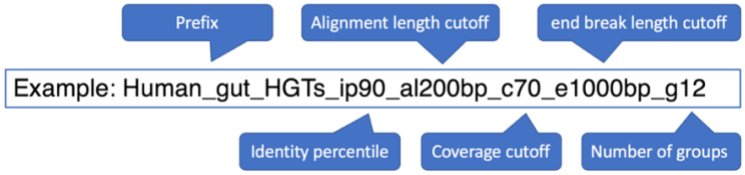
1. Grouping result is exported to **[prefix]_grouping_g[num]_[taxon_rank].txt**, which will be used as input for running Best-match.py and Phylogenetic.py.
2. Gene calling results in GenBank and FASTA format.
3. A phylogenetic tree of input genomes.
4. A bar plot shows the number of input genomes in each group at provided taxonomic rank.



3. Best-match.py

```
Best-match.py -h
-p                output prefix
-g                grouping file
-blastall         all vs all blast results
-cov              minimum coverage cutoff for blast hits, default is 70
-al               minimum alignment length cutoff for blast hits, default is 200
-flk              the length of flanking sequences to plot, default is 3000
-ip               identity percentile cutoff, default is 90
-eb               minimal length to be considered as at ends, default 500
-tmp              keep temporary files
-threads          number of threads, default is 1
```

Best-match.py will first perform an all versus all blastn among all predicted genes. You can provide BLAST results with option "`-blastall`", if you have it ready before hand. An example command for running blastn can be found below. Please make sure the output format is exactly the same (`-outfmt "6 qseqid sseqid pident length mismatch gapopen qstart qend sstart send evalue bitscore qlen slen"`). HGT candidates predicted by the best-match approach, as well as the plots of their flanking regions are exported to a folder named in the following format:



Example command:

```
# prepare blastdb
makeblastdb -in human_gut_combined.ffn -dbtype nucl -parse_seqids

# parameters for running blastn
blastn -query human_gut_combined.ffn -db human_gut_combined.ffn -out
human_gut_all_vs_all_ffn.tab -evalue 1e-5 -num_threads 12 -task blastn -outfmt "6 qseqid
sseqid pident length mismatch gapopen qstart qend sstart send evalue bitscore qlen slen"

# run Best-match.py
$ python Best-match.py -p human_gut -threads 6
```

Output files:

A list of HGT candidates identified by the BM 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**.

4. Phylogenetic.py

```
Phylogenetic.py -h
-p              output prefix
-o              orthologs folder
-t              SCG tree produced by PrepIn.py
-g              grouping file
-cov            coverage cutoff, default is 70
-al             alignment length cutoff, default is 200
-ip             identity percentile, default is 90
-eb             the minimal length to be considered as end break, default 500
-threads        number of threads, default is 1
```

All protein orthologs within the input genomes need to be provided for the phylogenetic approach, you can get them with GET_HOMOLOGUES [2]. The input file for GET_HOMOLOGUES is a folder (**[prefix]_gbk_files**) holds the annotation files of query genomes in Genbank format, which is one of the outputs from PrepIn.py. An example of the protein orthologs folder can be found at folder "`example_dataset`".

Example command:

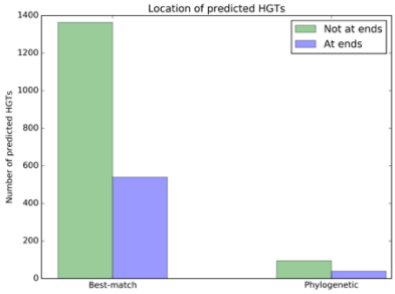
```
# run GET_HOMOLOGUES
$ get_homologues.pl -f 70 -t 3 -S 70 -E 1e-05 -C 70 -G -n 16 -X -d human_gut_gbk_files

# run Phylogenetic.py
$ python Phylogenetic.py -p human_gut -o human_gut_homologues -threads 6
```

Output files:

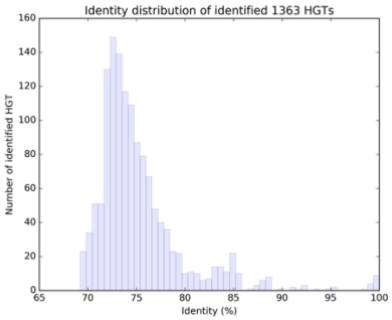
HGT candidates validated by PG approach are exported to the same folder as BM approach.

1. **HGT_candidates_PG.txt**: BM approach predicted HGTs, with additional information provided by the PG approach.
2. **HGT_candidates_PG_validated.txt**: HGTs that are only validated by the PG approach.
3. **HGT_candidates_PG_aa.fasta**: Nucleotide sequences of HGTs that are validated by the PG approach.
4. **HGT_candidates_PG_nc.fasta**: Amino acid sequences of HGTs that are validated by the PG approach.

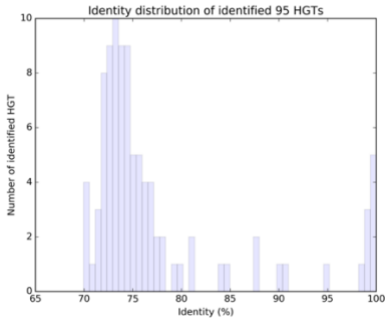
5. **[prefix]_plot_at_ends_stat.png**: Location statistics of predicted HGTs by BM and PG approaches.



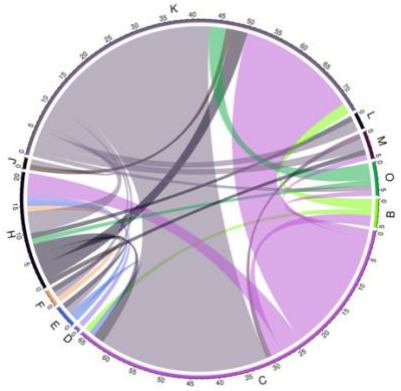
6. **[prefix]_plot_HGT_identities_BM.png**: Identity distribution of BM approach predicted HGTs.



7. **[prefix]_plot_HGT_identities_PG.png**: Identity distribution of predicted HGTs that are validated by the PG approach.



8. **[prefix]_plot_circos_PG.png**: Gene flow between groups. Bands on the plot connect donors and recipients, with the width of the band correlating to the number of HGTs and the colour corresponding to the donors.



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.