Command Line Center for Genomic Epidemiology

1) Downloading the CGE databases

In the following, please take care that unwanted newlines are not inserted into the command. This can happen if you copy/past a command that covers more than 1 line. In that case copy/paste the first line without hitting Enter and then copy/paste the next line directly after the first line.

KmerFinder

The KmerFinder database is a bit different than the other databases, since it due to its large size cannot be hosted by BitBucket. Accordingly, the data is located on an FTP server and can be downloaded using wget. First, make a folder called kmerfinder, then move into it. To download the database to the folder you are currently in type:

$ wget ftp://ftp.cbs.dtu.dk/public/CGE/databases/KmerFinder/version/20180706/bacteria.organisms.ATGAC.p

$ wget ftp://ftp.cbs.dtu.dk/public/CGE/databases/KmerFinder/version/20180706/bacteria.organisms.ATGAC.desc.p

$ wget ftp://ftp.cbs.dtu.dk/public/CGE/databases/KmerFinder/version/20180706/bacteria.organisms.ATGAC.len.p

$ wget ftp://ftp.cbs.dtu.dk/public/CGE/databases/KmerFinder/version/20180706/bacteria.organisms.ATGAC.ulen.p

Further, you need two files, called bacteria.tax and a config file, which can both be cloned from BitBucket:

$ git clone https://bitbucket.org/mettevoldbylarsen/kmerfinder_db-advanced-workshop/

The above command downloads the two files to a folder called kmerfinder_db-advanced-workshop. Move them to the kmerfinder folder (you have to stand in the kmerfinder folder, when you run the command):

$ mv kmerfinder_db-advanced-workshop/bacteria.tax .

$ mv kmerfinder_db-advanced-workshop/config .

FimTyper

Download the database to the folder you are currently in:

$ git clone https://bitbucket.org/genomicepidemiology/fimtyper_db

Rename database folder:

$ mv fimtyper_db fimtyper

MLST

Download the database to the folder you are currently in:

$ git clone https://bitbucket.org/genomicepidemiology/mlst_db

Rename database folder:

$ mv mlst_db mlst

pMLST

Download the database to the folder you are currently in:

$ git clone https://bitbucket.org/genomicepidemiology/pmlst_db

Rename database folder:

$ mv pmlst_db pmlst

PlasmidFinder

Download the database to the folder you are currently in:

$ git clone https://bitbucket.org/genomicepidemiology/plasmidfinder_db

Rename database folder:

$ mv plasmidfinder_db plasmidfinder

PointFinder

Download the database to the folder you are currently in:

$ git clone https://bitbucket.org/genomicepidemiology/pointfinder_db

Rename database folder:

$ mv pointfinder_db pointfinder

ResFinder

Download the database to the folder you are currently in:

$ git clone https://bitbucket.org/genomicepidemiology/resfinder_db

Rename database folder:

$ mv resfinder_db resfinder

SerotypeFinder

Download the database to the folder you are currently in:

$ git clone https://bitbucket.org/genomicepidemiology/serotypefinder_db

Rename database folder:

$ mv serotypefinder_db serotypefinder

The config file of the SerotypeFinder database needs to be modified slightly before it can be used with the SerotypeFinder Docker image later. While standing in the SerotypeFinder database folder, type the following:

$ head -3 config > headers.config
$ echo -e “H_type\tO_type\tecoli\tecoli serotype database” > end.config
$ cat headers.config end.config > config

VirulenceFinder

Download the database to the folder you are currently in:

$ git clone https://bitbucket.org/genomicepidemiology/virulencefinder_db/

Rename database folder:

$ mv virulencefinder_db virulencefinder

2) Running the CGE methods via Docker images

KmerFinder

Note: Unlike the other -Finder methods, KmerFinder requires more RAM than what is available on a t2.micro. Choose a larger instance type and also choose more storage than the default 8 GB, as the KmerFinder DB on its own is quite large.

First pull the Docker image:

$ docker pull goseqit/kmerfinder_goseqit_docker

Next, the command for running the method inside the Docker container:

$ docker run -ti --rm -w /output -v ~/[database folder]:/databases -v ~/[folder with input data]:/input -v ~/[folder to write output files to]:/output goseqit/kmerfinder_goseqit_docker KmerFinder -f /input/[file.fsa] > [folder to write output files to]/log_KmerFinder

FimTyper

First pull the Docker image:

$ docker pull goseqit/fimtyper_goseqit_docker

Next, the command for running the method inside the Docker container:

$ docker run -ti --rm -w /output -v ~/[database folder]:/databases -v ~/[folder with input data]:/input -v ~/[folder to write output files to]:/output goseqit/fimtyper_goseqit_docker FimTyper -f /input/[file.fsa] -k 95 > [folder to write output files to]/log_FimTyper

Where:
-k: Threshold for min. %ID. An integer between 50-100. Default is 95.

pMLST

First pull the Docker image:

$ docker pull goseqit/pmlst_goseqit_docker

Next, the command for running the method inside the Docker container:

$ docker run -ti --rm -w /output -v ~/[database folder]:/databases -v ~/[folder with input data]:/input -v ~/[folder to write output files to]:/output goseqit/pmlst_goseqit_docker pMLST -f /input/[file.fsa] -s [pMLST scheme] > [folder to write output files to]/log_pmlst

Where:
-s: Used for specifying which pMLST scheme to use. See config file in pMLST database for schemes to choose among. Multiple schemes are separated with comma (e.g., “-s incn,inchi1,inci1”).

PlasmidFinder

First pull the Docker image:

$ docker pull goseqit/plasmidfinder_goseqit_docker

Next, the command for running the method inside the Docker container:

$ docker run -ti --rm -w /output -v ~/[database folder]:/databases -v ~/[folder with input data]:/input -v ~/[folder to write output files to]:/output goseqit/plasmidfinder_goseqit_docker PlasmidFinder -f /input/[inputfile.fsa] -s [sub_database] -k [%ID] > ~/[path and folder to write output files to]/log_PlasmidFinder

Where:
-k: Threshold for min. %ID. An integer between 50-100. Default is 90.
-s: Used for specifying which PlasmidFinder subdatabase to use. See config file in PlasmidFinder database for schemes to choose among. Multiple schemes are separated with comma (e.g., "-s gram_positive,enterobacteriaceae").

MLST

First pull the Docker image:

$ docker pull goseqit/mlst-advanced-workshop-docker

Next, the command for running the method inside the Docker container:

$ docker run -ti --rm -w /output -v ~/[database folder]:/databases -v ~/[folder with input data]:/input -v ~/[folder to write output files to]:/output goseqit/mlst_advanced_workshop_docker MLST -f /input/[inputfile.fsa] -s [mlst scheme] > ~/[path and folder to write output files to]/log_MLST

-s [scheme]: Used for specifying which MLST scheme to use. In the config file in the MLST database, you can see all the possible MLST schemes to choose among. Multiple schemes are separated with comma (e.g., "-s ecoli,ecoli_2").

PointFinder (looks for point mutations in chromosomal genes that cause antimicrobial resistance)

First pull the Docker image:

$ docker pull goseqit/pointfinder_goseqit_docker

Next, the command for running the method inside the Docker container:

$ docker run -ti --rm -w /output -v ~/[database folder]:/databases -v ~/[folder with input data]:/input -v ~/[folder to write output files to]:/output goseqit/pointfinder_goseqit_docker PointFinder -f /input/[inputfile.fsa] -s [species] -k 90 -l 0.5 > ~/[path and folder to write output files to]/log_pointfinder

Where:
-s: Used for specifying which species to look for point mutations in. Multiple species are separated by comma, e.g., “e.coli,salmonella”.
-k: Used for specifying minimum %Identity between allele in PointFinder database and sequence in input genome (integer between 50 - 100). Default is 90.
-l: Used for specifying minimum coverage of allele in PointFinder database by sequence in input genome (floating point between 0.5 - 1.0). Default is 0.5.

ResFinder (looks for acquired genes that cause antimicrobial resistance)

First pull the Docker image:

$ docker pull goseqit/resfinder_goseqit_docker

Next, the command for running the method inside the Docker container:

$ docker run -ti --rm -w /output -v ~/[database folder]:/databases -v ~/[folder with input data]:/input -v ~/[folder to write output files to]:/output goseqit/resfinder_goseqit_docker ResFinder -f /input/[inputfile.fsa] -s [group of antimicrobials] -k 90 -l 0.5 > ~/[path and folder to write output files to]/log_resfinder

Where:

-s: Used for specifying which group of antimicrobials to search for resistance genes towards. See the config file of the ResFinder database for groups to choose among. Multiple groups are separated by comma, e.g., “aminoglycoside,beta-lactam,tetracycline”.
-k: Used for specifying minimum %Identity between allele in ResFinder database and sequence in input genome (integer between 50 - 100). Default is 90.
-l: Used for specifying minimum coverage of allele in ResFinder database by sequence in input genome (floating point between 0.5 - 1.0). Default is 0.5.

SerotypeFinder

First pull the Docker image:

$ docker pull goseqit/serotypefinder_goseqit_docker

Next, the command for running the method inside the Docker container:

$ docker run -ti --rm -w /output -v ~/[database folder]:/databases -v ~/[folder with input data]:/input -v ~/[folder to write output files to]:/output goseqit/serotypefinder_goseqit_docker SerotypeFinder -f /input/[inputfile.fsa] -k 90 -l 0.5 > ~/[path and folder to write output files to]/log_serotypefinder

Where:
-k: Used for specifying minimum %Identity between allele in SerotypeFinder database and sequence in input genome (integer between 50 - 100). Default is 90.
-l: Used for specifying minimum coverage of allele in SerotypeFinder database by sequence in input genome (floating point between 0.5 - 1.0). Default is 0.5.

VirulenceFinder

First pull the Docker image:

$ docker pull goseqit/virulencefinder_goseqit_docker

Next, the command for running the method inside the Docker container:

$ docker run -ti --rm -w /output -v ~/[database folder]:/databases -v ~/[folder with input data]:/input -v ~/[folder to write output files to]:/output goseqit/virulencefinder_goseqit_docker VirulenceFinder -f /input/[inputfile.fsa] -s [sub database] -k 95 > ~/[path and folder to write output files to]/log_virulencefinder

Where:
-s: Used for specifying sub database(s). See the config file for available sub databases. Multiple sub databases are seperated by commas.
-k: Used for specifying minimum %Identity between allele in VirulenceFinder database and sequence in input genome (integer between 50 - 100). Default is 95.

The Bacterial Analysis Pipeline

Note: The Bacterial Analysis Pipeline requires more RAM than what is available on a t2.micro instance. Choose a larger instance type and also choose more storage than the default 8 GB.

If you want to run the Bacterial Analysis Pipeline command line, you first have to download the databases for KmerFinder, MLST, ResFinder, PlasmidFinder, pMLST, and VirulenceFinder. In your home directory, create a folder called “databases_all” with sub-folders called “kmerfinder”, “mlst”, “resfinder", “plasmidfinder”, “pmlst”, “virulencefinder” (see above how to download each of the databases).

Now pull the Docker image:

$ docker pull goseqit/bap-goseqittools

Next, the command for running the pipeline inside the Docker container:

$ docker run -ti --rm -w /output -v ~/databases_all:/databases -v ~/[folder with input data]:/workdir goseqit/bap-goseqittools BAP --wdir /workdir --fa [inputfile.fsa] > ~/[path and folder to write output files to]/log_bap

Note that you can only specify one directory, workdir, which should contain the inputfile and to which the output files will also be written. For more documentation, see: https://bitbucket.org/mettevoldbylarsen/bap-goseqittools/src/master/.

3) Running a BLAST-based method with a customised database

Many of the above methods are build up the same way with a database of genes in fasta format that are searched by BLAST. If you have your own database of genes in fasta format, you can easily trick, e.g., ResFinder, to search it. Just follow these steps.

1) Create your database of genes in fasta format and with the extension .fsa. Note: the database must contain genes - that is DNA (A,T,C,G), and not protein sequences. Make sure each fasta header has a unique identifier.

Example:
Filename: MyDB.fsa
Content (you can add as many genes as you would like):

```
>dsbD
ATGGCTCAACGCATCTTTACGCTGATCCTGCTACTTTGCAGCACTTCCGTTTTTGCCGGATTATTCGACG
CGCCGGGACGTTCACAATTTGTCCCCGCGGATCAAGCCTTTGCTTTTGATTTTCAGCAAAACCAACATGA
CCTTAATCTGACCTGGCAGATCAAAGACGGTTACTACCTCTACCGTAAACAGATCCGCATTACGCCGGAA
CACGCGAAAATTGCCGACGTGCAGCTGCCGCAAGGCGTCTGGCATGAAGATGAGTTTTACGGCAAAAGCG
AGATTTACCGCGATCGGCTGACGCTTCCCGTCACCATCAACCAGGCGAGTGCGGGAGCGACGTTAACTGT
CACCTACCAGGGCTGTGCTGATGCCGGTTTCTGTTATCCGCCAGAAACCAAAACCGTTCCGTTAAGCGAA
GTGGTCGCCAACAACGCAGCGCCACAGCCTGTGTCTGTTCCGCAGCAAGAGCAGCCCACCGCGCAATTGC
CCTTTTCCGCGCTCTGGGCGTTGTTGATCGGTATTGGTATCGCCTTTACGCCATGCGTGCTGCCAATGTA
CCCACTGATTTCTGGCATCGTGCTGGGTGGTAAACAGCGGCTCTCCACTGCCAGAGCATTGTTGCTGACC
TTTATTTATGTGCAGGGGATGGCGCTGACCTACACGGCGCTGGGTCTGGTGGTTGCCGCCGCAGGGTTAC
AGTTCCAGGCGGCGCTACAGCACCCATACGTGCTCATTGGCCTCGCCATCGTCTTTACCTTGCTGGCGAT
GTCAATGTTTGGCTTGTTTACCCTGCAACTCCCCTCTTCGCTGCAAACACGTCTCACGTTGATGAGCAAT
CGCCAACAGGGCGGCTCACCTGGCGGTGTGTTTGTTATGGGGGCGATTGCCGGACTGATCTGTTCACCAT
GCACCACCGCACCGCTTAGCGCGATTCTGCTGTATATCGCCCAAAGCGGGAACATGTGGCTGGGCGGCGG
CACGCTTTATCTCTATGCGTTGGGCATGGGCCTGCCGCTGATGCTAATTACCGTCTTTGGTAACCGCTTG
CTGCCGAAAAGCGGCCCGTGGATGGAACAAGTCAAAACCGCGTTTGGTTTTGTGATCCTCGCACTGCCGG
TCTTCCTGCTGGAGCGAGTGATTGGTGATGTATGGGGATTACGCTTGTGGTCGGCGCTGGGTGTCGCATT
CTTTGGCTGGGCCTTTATCACCAGCCTACAGGCTAAACGCGGCTGGATGCGTATTGTGCAAATTATTCTG
CTGGCAGCGGCATTGGTTAGCGTGCGCCCACTTCAGGATTGGGCATTTGGTGCGACGCATACCGCGCAAA
CTCAGACGCATCTCAACTTTACACAAATCAAAACGGTAGATGAGTTAAATCAGGCGCTCGTTGAAGCCAA
AGGCAAACCGGTGATGTTAGATCTTTATGCCGACTGGTGCGTCGCCTGTAAAGAGTTTGAGAAATACACC
TTCAGCGACCCGCAGGTGCAAAAAGCGTTAGCAGACACGGTCTTACTTCAGGCCAACGTCACGGCCAACG
ACGCACAAGATGTGGCGCTGTTAAAGCATCTTAATGTCCTTGGCCTACCGACAATTCTCTTTTTTGACGG
ACAAGGCCAGGAGCATCCACAAGCACGCGTCACGGGCTTTATGGATGCTGAAACCTTCAGCGCACATTTG
CGCGATCGCCAACCGTGA
>PapG
ATGAAAAAATGGTTCCCAGCTTTGTTATTTTCCTTGTGTGTGTCTGGTGAGTCCTCTGCA
```

```
TGGAATAATATTGTCTTTTACTCCCTTGGAGACGTTAACTCTTATCAGGGAGGGAATGTG
GTGATTACTCAAAGGCCACAATTTATAACTTCGTGGCGCCCGGGCATTGCTACGGTAACC
TGGAATCAGTGTAATGGTCCTGGGTTTGCTGATGGTTTCTGGGCTTACTACAGGGAGTAT
ATTGCGTGGGTAGTATTCCCCAAAAAGGTTATGACCCAAAATGGATATCCCTTATTTATT
GAGGTTCATAATAAAGGTAGCTGGAGTGAGGAGAATACTGGTGACAATGACAGCTATTTT
TTTCTCAAGGGGTATAAGTGGGATGAGCGGGCCTTTGATACAGCTAATTTGTGTCAGAAA
CCAGGAGAAAAAACCCGTCTGACTGAGAAATTTGACGATATTATTTTTAAAGTCGCCCTA
CCTGCAGATCTTCCTTTAGGGGATTATTCTGTTACAATTCCATACACTTCCGGCATGCAG
CGTCATTTCGCGAGTTACTTGGGGGCCCGTTTTAAAATCCCATACAATGTGGCCAAAACT
CTCCCAAGAGAGAATGAAATGTTATTCTTATTTAAGAATATCGGCGGATGCCGTCCTTCT
GCACAGTCTCTGGAAATAAAGCATGGTGATCTGTCTATTAATAGCGCTAATAATCATTAT
GCGGCTCAGACTCTTTCTGTGTCTTGCGATGTGCCTGCAAATATTCGTTTTATGCTGTTA
AGAAATACAGCTCCGACATACAGCCATGGTAAGAAATTTTCGGTTGGTCTGGGTCATGGC
TGGGACTCCATTGTTTCGGTTAACGGGGTGGACACAGGAGAGACAACGATGAGATGGTAC
AAAGCAGGTACACAAAACCTGACCATCGGCAGTCGCCTCTATGGTGAATCTTCAAAGATA
CAACCAGGAGTACTATCTGGTTCAGCAACGCTGCTCATGATATTGCCATAA
```

Note: Remember to safe your database using a clean text editor like Sublime Text, and not, e.g., Word, which will safe a lot of formatting information along with the gene sequence.

2) While standing in the ResFinder database folder, add the database to the ResFinder config file:

$ echo -e “MyDB\tMy own database\tSome additional text\n” >> config

#Note it is only important that the first element of the line, here “MyDB” is identical to the name of your database, minus the extension. It doesn’t matter what you write as second and third element.

3) Run ResFinder via the Docker image as usual specifying the database you just added:

$ docker run -ti --rm -w /output -v ~/[database folder]:/databases -v ~/[folder with input data]:/input -v ~/[folder to write output files to]:/output goseqit/resfinder_goseqit_docker ResFinder -f /input/[inputfile.fsa] **-s MyDB** -k 90 -l 0.5 > ~/[path and folder to write output files to]/log_resfinder

When the run has finished you can find the output files in the output folder as usual.