Cortex cheat sheet for bacterial genomics

1 Requirements

Download Cortex from github (https://github.com/iqbal-lab/cortex) as follows:

```
git clone --recursive https://github.com/iqbal-lab/cortex.git
```

You must have installed VCFtools (and have the entire directory, not just the binary executable), R and Stampy. R must be in your path. Add the following directories to PERL5LIB and PATH

```
export PERL5LIB= /path/cortex/scripts/analyse_variants/
    bioinf-perl/lib;/path/cortex/scripts/calling:$PERL5LIB
export PATH = /path/cortex/scripts/analyse_variants/
   needleman_wunsch-0.3.0
```

Compile and install cortex for 1,2,N,N+1 colours where N=number of samples

2 Preparation (once per species)

Assuming you want to report results with coordinates on a reference genome:

- Build a Stampy hash of the genome

```
stampy.py -G stampy_refhash ref.fa
stampy.py -g stampy_refhash -H stampy_refhash
```

- Build a Cortex binary graph file of the reference genome at the kmer size you are going to use (start with k=31). Don't worry about the –max_read_len 10000 argument, for reference fasta this just means the file is parsed in chunks of size 10000bp.

```
ls /path/to/ref.fasta > ref_fa_list
cortex_var_31_c1 --kmer_size 31  --mem_height 17
   --mem_width 100  --se_list ref_fa_list
   --max_read_len 10000  --dump_binary ref.k31.ctx
   --sample_id REF
```

- Make an INDEX file which maps each sample to a LIST of fastq (or bams)

3 Compare each sample against a reference independently

Step1 First we build per-sample graphs in parallel using commands such as this GNU parallels command. Within directory OUTDIR1, we will make a directory for each sample. A typical command would be (for 1700 samples):

```
parallel --gnu -j 20  perl scripts/calling/par.pl
  --num {}  --index INDEX   --list_ref LISTREF
  --refbindir DIR_WITH_CTX_BINS --index_dir indexes/
  --stampy_bin stampy-1.0.23/stampy.py
  --stampy_hash  stampy_refhash  --bc yes  --pd no
  --kmer 31 --mem_height 18  --mem_width 100
  --qthresh 10  --cortex_dir /path/to/cortex/
  --out_dir OUTDIR1  --vcftools_dir /path/vcftools_0.1.9/
    ::: {1..1700}
```

Step2 Combine all the per-sample VCFs to get one combined set of sites (SNPs, indels, structural variants).

```
perl scripts/analyse_variants/combine/combine_vcfs.pl
  --list_vcfs list_all_raw_vcfs  --vcftools_dir vcftools_0.1.9
  --outdir OUTDIR2  --prefix presidents  --refname REF_v1
  --ref_fasta ref.fa --rootdir_for_sample_output  OUTDIR1
  --kmer 31 --mem_height 18 --mem_width 150
   --ref_binary  ref.k31.ctx
```

Step 3 Finally, independently genotype each sample at all of these sites. Here using GNU parallels to spread across 20 cores of a server. Memory use now drops as we only use the graph of poymorphisms.

```
cat OUTDIR2/list_args_for_final_step | parallel --colsep '\t'
     perl scripts/calling/genotype_1sample_against_sites.pl
 --config OUTDIR2/config.txt
  --invcf OUTDIR2/presidents.sites_vcf
--sample {1} --outdir {2} --genome_size 24000000
  --sample_graph {3} --mem_height 17  --mem_width 100
```

This will give you one VCF file per sample in OUTDIR1/{sample_id}/union_calls/

4 Segregating variants within our dataset (joint workflow)

First build sample graphs as Step1 in the previous example. Then UNDETERMINED

5 Pan-genome analysis

To detect presence of a set of predefined genes (genes.fasta) among your samples

To look at pan-genome graph of all samples and see which samples have which contigs, allowing you to stratify them by frequency or look for differentiating/segregating contigs.

6 The End

For further information: