Running digital normalization and artifact removal on an HMP mock data set

This notebook runs 'digital normalization' (see http://ged.msu.edu/papers/2012-diginorm/) and Illumina artifact removal on an HMP Illumina mock data set. The dataset and description originated from http://www.ncbi.nlm.nih.gov/bioproject/48475.

Prerequisites for this tutorial

I. Start a new NGS 2012 EC2 instance http://ged.msu.edu/angus/tutorials-2012/start-up-an-ec2-instance.html
II. Install khmer and screed In your ipython notebook homepage: run ngs-00-update-notebooks and ngs-03-install-khmer
III. Make a volume of a snapshot (snap-08efea77) containing the HMP data and mount it on your instance as /hmp-mock-tutorial

We are going to start with the HMP gzipped fastq Illumina sequencing reads and:

I. Normalize to coverage = 10
II. Trim high-abundance (likely Illumina artifacts)

This produces files that can be used as input into the partitioning algorithm (see the next notebook!)

In [2]:
```
cd /test/raw-data
```

```
/test/raw-data
```

In [3]:
```
!ls
```

```
SRR172902.fastq.gz  SRR172903.fastq.gz
```

We are going to be working with the SRR172903 dataset (staggered mixture). You can play with a combination of the two or the other dataset (even mixture) later if you like.

Pass 1: normalize to C=10.

The only parameter to change here is the memory, which is fixed at 4gb (multiply the -N and -x parameters).

This should take 10-15 minutes.

In [4]:
```
cd /test
```

```
/test
```

In [5]:
```
mkdir tutorial-files
```

In [6]:
```
cd /test/tutorial-files
```

```
/test/tutorial-files
```

Now, let's run the digital normalization for a word length of -k 20, we are going to remove redundant reads which are contribute to a coverage greater than -C 10. We'll be using a hashtable size of -x 1e9 and four of them -N 4. We'll save this hashtable as mock-pass1.kh. The actual size of your hashtable is dependent on the characteristics of your dataset - the important thing is that the false positive rate in the hashtable (caluclated at the end) is less than 15%. The output will be saved in pass1.report. The input of reads is the last parameter of the command (there can be multiple read files here). The output shows you the number of reads kept over the total of reads processed.

In [7]:
```
!python /usr/local/src/khmer/scripts/normalize-by-median.py -k 20 -C 10 -N
```

```
PARAMETERS:
 - kmer size =    20            (-k)
 - n hashes =     4             (-N)
 - min hashsize = 1e+09         (-x)
 - paired =           False             (-p)

Estimated memory usage is 4e+09 bytes (n_hashes x min_hashsize)
--------
making hashtable
... kept 98471 of 100000 , or 98 %
... in file /test/raw-data/SRR172903.fastq.gz
... kept 194971 of 200000 , or 97 %
... in file /test/raw-data/SRR172903.fastq.gz
... kept 289555 of 300000 , or 96 %
... in file /test/raw-data/SRR172903.fastq.gz
... kept 383133 of 400000 , or 95 %
... in file /test/raw-data/SRR172903.fastq.gz
... kept 476015 of 500000 , or 95 %
... in file /test/raw-data/SRR172903.fastq.gz
... kept 568701 of 600000 , or 94 %
... in file /test/raw-data/SRR172903.fastq.gz
... kept 661095 of 700000 , or 94 %
... in file /test/raw-data/SRR172903.fastq.gz
... kept 753224 of 800000 , or 94 %
... in file /test/raw-data/SRR172903.fastq.gz
... kept 845404 of 900000 , or 93 %
... in file /test/raw-data/SRR172903.fastq.gz
... kept 937527 of 1000000 , or 93 %
... in file /test/raw-data/SRR172903.fastq.gz
... kept 1030022 of 1100000 , or 93 %
... in file /test/raw-data/SRR172903.fastq.gz
... kept 1121084 of 1200000 , or 93 %
... in file /test/raw-data/SRR172903.fastq.gz
... kept 1211308 of 1300000 , or 93 %
... in file /test/raw-data/SRR172903.fastq.gz
... kept 1301535 of 1400000 , or 92 %
... in file /test/raw-data/SRR172903.fastq.gz
... kept 1391544 of 1500000 , or 92 %
... in file /test/raw-data/SRR172903.fastq.gz
... kept 1480364 of 1600000 , or 92 %
... in file /test/raw-data/SRR172903.fastq.gz
```

```
... kept 1568191 of 1700000 , or 92 %
... in file /test/raw-data/SRR172903.fastq.gz
... kept 1655168 of 1800000 , or 91 %
... in file /test/raw-data/SRR172903.fastq.gz
... kept 1743478 of 1900000 , or 91 %
... in file /test/raw-data/SRR172903.fastq.gz
... kept 1828559 of 2000000 , or 91 %
... in file /test/raw-data/SRR172903.fastq.gz
... kept 1912000 of 2100000 , or 91 %
... in file /test/raw-data/SRR172903.fastq.gz
... kept 1997606 of 2200000 , or 90 %
... in file /test/raw-data/SRR172903.fastq.gz
... kept 2080707 of 2300000 , or 90 %
... in file /test/raw-data/SRR172903.fastq.gz
... kept 2164877 of 2400000 , or 90 %
... in file /test/raw-data/SRR172903.fastq.gz
... kept 2247921 of 2500000 , or 89 %
... in file /test/raw-data/SRR172903.fastq.gz
... kept 2324387 of 2600000 , or 89 %
... in file /test/raw-data/SRR172903.fastq.gz
... kept 2398885 of 2700000 , or 88 %
... in file /test/raw-data/SRR172903.fastq.gz
... kept 2471525 of 2800000 , or 88 %
... in file /test/raw-data/SRR172903.fastq.gz
... kept 2543473 of 2900000 , or 87 %
... in file /test/raw-data/SRR172903.fastq.gz
... kept 2621185 of 3000000 , or 87 %
... in file /test/raw-data/SRR172903.fastq.gz
... kept 2691512 of 3100000 , or 86 %
... in file /test/raw-data/SRR172903.fastq.gz
... kept 2771750 of 3200000 , or 86 %
... in file /test/raw-data/SRR172903.fastq.gz
... kept 2841597 of 3300000 , or 86 %
... in file /test/raw-data/SRR172903.fastq.gz
... kept 2912094 of 3400000 , or 85 %
... in file /test/raw-data/SRR172903.fastq.gz
... kept 2976001 of 3500000 , or 85 %
... in file /test/raw-data/SRR172903.fastq.gz
... kept 3038330 of 3600000 , or 84 %
... in file /test/raw-data/SRR172903.fastq.gz
... kept 3099250 of 3700000 , or 83 %
... in file /test/raw-data/SRR172903.fastq.gz
... kept 3156056 of 3800000 , or 83 %
... in file /test/raw-data/SRR172903.fastq.gz
... kept 3212197 of 3900000 , or 82 %
... in file /test/raw-data/SRR172903.fastq.gz
... kept 3267587 of 4000000 , or 81 %
... in file /test/raw-data/SRR172903.fastq.gz
... kept 3320406 of 4100000 , or 80 %
... in file /test/raw-data/SRR172903.fastq.gz
... kept 3371773 of 4200000 , or 80 %
... in file /test/raw-data/SRR172903.fastq.gz
... kept 3422626 of 4300000 , or 79 %
... in file /test/raw-data/SRR172903.fastq.gz
```

```
... kept 3471702 of 4400000 , or 78 %
... in file /test/raw-data/SRR172903.fastq.gz
... kept 3520891 of 4500000 , or 78 %
... in file /test/raw-data/SRR172903.fastq.gz
... kept 3580270 of 4600000 , or 77 %
... in file /test/raw-data/SRR172903.fastq.gz
... kept 3632578 of 4700000 , or 77 %
... in file /test/raw-data/SRR172903.fastq.gz
... kept 3681283 of 4800000 , or 76 %
... in file /test/raw-data/SRR172903.fastq.gz
... kept 3737769 of 4900000 , or 76 %
... in file /test/raw-data/SRR172903.fastq.gz
... kept 3796621 of 5000000 , or 75 %
... in file /test/raw-data/SRR172903.fastq.gz
... kept 3841452 of 5100000 , or 75 %
... in file /test/raw-data/SRR172903.fastq.gz
... kept 3883343 of 5200000 , or 74 %
... in file /test/raw-data/SRR172903.fastq.gz
... kept 3926327 of 5300000 , or 74 %
... in file /test/raw-data/SRR172903.fastq.gz
... kept 3967557 of 5400000 , or 73 %
... in file /test/raw-data/SRR172903.fastq.gz
... kept 4007282 of 5500000 , or 72 %
... in file /test/raw-data/SRR172903.fastq.gz
... kept 4046391 of 5600000 , or 72 %
... in file /test/raw-data/SRR172903.fastq.gz
... kept 4084792 of 5700000 , or 71 %
... in file /test/raw-data/SRR172903.fastq.gz
... kept 4122792 of 5800000 , or 71 %
... in file /test/raw-data/SRR172903.fastq.gz
... kept 4159952 of 5900000 , or 70 %
... in file /test/raw-data/SRR172903.fastq.gz
... kept 4196225 of 6000000 , or 69 %
... in file /test/raw-data/SRR172903.fastq.gz
... kept 4231920 of 6100000 , or 69 %
... in file /test/raw-data/SRR172903.fastq.gz
... kept 4267195 of 6200000 , or 68 %
... in file /test/raw-data/SRR172903.fastq.gz
... kept 4301540 of 6300000 , or 68 %
... in file /test/raw-data/SRR172903.fastq.gz
... kept 4335437 of 6400000 , or 67 %
... in file /test/raw-data/SRR172903.fastq.gz
... kept 4369518 of 6500000 , or 67 %
... in file /test/raw-data/SRR172903.fastq.gz
... kept 4402650 of 6600000 , or 66 %
... in file /test/raw-data/SRR172903.fastq.gz
... kept 4434925 of 6700000 , or 66 %
... in file /test/raw-data/SRR172903.fastq.gz
... kept 4467070 of 6800000 , or 65 %
... in file /test/raw-data/SRR172903.fastq.gz
... kept 4499248 of 6900000 , or 65 %
... in file /test/raw-data/SRR172903.fastq.gz
... kept 4531198 of 7000000 , or 64 %
... in file /test/raw-data/SRR172903.fastq.gz
```

```
... kept 4562298 of 7100000 , or 64 %
... in file /test/raw-data/SRR172903.fastq.gz
... kept 4593518 of 7200000 , or 63 %
... in file /test/raw-data/SRR172903.fastq.gz
... kept 4624607 of 7300000 , or 63 %
... in file /test/raw-data/SRR172903.fastq.gz
... kept 4655495 of 7400000 , or 62 %
... in file /test/raw-data/SRR172903.fastq.gz
... kept 4686324 of 7500000 , or 62 %
... in file /test/raw-data/SRR172903.fastq.gz
... kept 4717418 of 7600000 , or 62 %
... in file /test/raw-data/SRR172903.fastq.gz
... kept 4747864 of 7700000 , or 61 %
... in file /test/raw-data/SRR172903.fastq.gz
... kept 4780874 of 7800000 , or 61 %
... in file /test/raw-data/SRR172903.fastq.gz
... kept 4813815 of 7900000 , or 60 %
... in file /test/raw-data/SRR172903.fastq.gz
DONE with /test/raw-data/SRR172903.fastq.gz ; kept 4823693 of 7932819 or
60 %
output in SRR172903.fastq.gz.keep
Saving hashfile through /test/raw-data/SRR172903.fastq.gz
...saving to /test/tutorial-files/mock-pass1.kh
fp rate estimated to be 0.000
```

Check out the fp rate. We're good to go... What output files were produced?

In [8]:
```
!ls
```

```
mock-pass1.kh  pass1.report  SRR172903.fastq.gz.keep
```

In [9]:
```
!head pass1.report
```

```
100000 98471 0.98471
200000 194971 0.974855
300000 289555 0.965183333333
400000 383133 0.9578325
500000 476015 0.95203
600000 568701 0.947835
700000 661095 0.944421428571
800000 753224 0.94153
900000 845404 0.939337777778
1000000 937527 0.937527
```

You can make a graph of the rate at which diginorm eliminates reads. What does this tell you about the diversity (or redundancy of your dataset?)

Pass 2: Trim off high-abundance k-mers

These are likely to be the results of Illumina crap. Partitioning (what we do next) is much harder if you don't do this, and the assemblies may be less good. We'll be trimming sequences where we find k-mers which are present in our dataset at a coverage greater than 50. Note, that we should not do this without

doing digital normalization first, why?

This should take ~10-15 minutes.

In [10]:
```
!cd /test/tutorial-files/
```

In [11]:
```
!python /usr/local/src/khmer/sandbox/filter-below-abund.py /test/tutorial-
```

```
file with ht: /test/tutorial-files/mock-pass1.kh
-- settings:
N THREADS 8
--
making hashtable
filtering /test/tutorial-files/SRR172903.fastq.gz.keep
starting threads
starting writer
loading...
... filtering 0
... filtering 100000
... filtering 200000
... filtering 300000
... filtering 400000
... filtering 500000
processed 500000 / wrote 497929 / removed 2071
processed 37552500 bp / wrote 36330097 bp / removed 1222403 bp
discarded 3.3%
... filtering 600000
... filtering 700000
... filtering 800000
... filtering 900000
... filtering 1000000
processed 1000000 / wrote 996526 / removed 3474
processed 75120000 bp / wrote 72750120 bp / removed 2369880 bp
discarded 3.2%
... filtering 1100000
... filtering 1200000
... filtering 1300000
... filtering 1400000
... filtering 1500000
processed 1500000 / wrote 1468336 / removed 31664
processed 112627500 bp / wrote 107250936 bp / removed 5376564 bp
discarded 4.8%
... filtering 1600000
... filtering 1700000
... filtering 1800000
... filtering 1900000
... filtering 2000000
processed 2000000 / wrote 1930381 / removed 69619
processed 150120000 bp / wrote 141053785 bp / removed 9066215 bp
discarded 6.0%
... filtering 2100000
```

```
... filtering 2200000
... filtering 2300000
... filtering 2400000
... filtering 2500000
processed 2500000 / wrote 2405056 / removed 94944
processed 187635000 bp / wrote 175741929 bp / removed 11893071 bp
discarded 6.3%
... filtering 2600000
... filtering 2700000
... filtering 2800000
... filtering 2900000
... filtering 3000000
processed 3000000 / wrote 2847941 / removed 152059
processed 225142500 bp / wrote 208178468 bp / removed 16964032 bp
discarded 7.5%
... filtering 3100000
... filtering 3200000
... filtering 3300000
... filtering 3400000
... filtering 3500000
processed 3500000 / wrote 3344495 / removed 155505
processed 262650000 bp / wrote 244556162 bp / removed 18093838 bp
discarded 6.9%
... filtering 3600000
... filtering 3700000
... filtering 3800000
... filtering 3900000
... filtering 4000000
processed 4000000 / wrote 3803586 / removed 196414
processed 300150000 bp / wrote 278243936 bp / removed 21906064 bp
discarded 7.3%
... filtering 4100000
... filtering 4200000
... filtering 4300000
... filtering 4400000
... filtering 4500000
processed 4500000 / wrote 4299172 / removed 200828
processed 337620000 bp / wrote 314547692 bp / removed 23072308 bp
discarded 6.8%
... filtering 4600000
... filtering 4700000
... filtering 4800000
done loading in sequences
DONE writing.
processed 4817893 / wrote 4614127 / removed 203766
processed 361476975 bp / wrote 337614355 bp / removed 23862620 bp
discarded 6.6%
```

This needs to be fixed later but for some reason our output filtered file is being saved somewhere else...we'll fix this later but for now let's just go move it to where I want it.

In [12]:
```
cd /test/tutorial-files/
```

```
/test/tutorial-files
```

In [13]:
```
ls
```

```
mock-pass1.kh  SRR172903.fastq.gz.keep
pass1.report   SRR172903.fastq.gz.keep.below
```

Okay, now we have a set of reads which are can be assembled -- we've normalized the coverage and removed artifacts. Sometimes, you may have a dataset that is too large to assemble on your computational resources. If this is the case, you can partition your reads by connectivity. That is in the next notebook, hmp-partitioning.

In [13]: