Friday 14th December, morning session: De Novo Assembly

Data sets

All data is derived from the GAGE website:
http://gage.cbcb.umd.edu/index.html
You can also check out the accompanying paper (Salzberg et al., 2011) published in Genome Research:
http://genome.cshlp.org/content/early/2012/01/12/gr.131383.111

Data sets can be found at:
http://gage.cbcb.umd.edu/data/index.html

All the test datasets that you will need for the practical session of this morning are already included in the WageningenEU AMI that you are using, no need therefore to upload the data yourself. The data can be found in the directory "GAGE_WageningenEU_repo", which is placed in your home directory (/home/ubuntu). In that directory, you will find two more directories, called 'Staph', which contains what is needed for the assembly of the *Staphylococcus aureus* genome, and 'HChr14', which contains what is need for the assembly of human chromosome 14.

k-mer based error corrections

Data preparation and filtering, and error correction may be one of the most critical steps in achieving a good assembly. In the words of Salzberg et al.(2011):

"One of the most important steps in any assembly, often taking much longer than the assembly itself, is the data cleaning process. WGS data are never perfect, and the various types of errors can cause different problems for different assemblers. High-quality data can produce dramatic differences in the results: for example, one assembly of the Rhodobacter sphaeroides *data had a contig N50 size of just 233 bp, but after error correction the same assembler achieved a contig N50 of 7793 bp, more than 30 times larger".*

Since the k-mer based corrections by Quake are quite computation time and memory intensive, we have chosen to provide the corrected dataset instead of the original, raw data. This is the data you will use for the assembly. The data can be found in a directory "Data/quakeCor/", in each of the species-specific directories. For more information on k-mer based error correction using Quake:
http://www.cbcb.umd.edu/software/quake/

Note that several assemblers, including SOAPdenovo, have built-in error k-mer based error corrections. We will nevertheless use the Quake-corrected dataset, because Salzberg et al. (2011) identified that even those assemblers with built-in correction algorithms do produce better assemblies even pre-corrected data.

SOAPdenovo

Because of time constraints, we will only evaluate one assembler this morning. We have chosen SOAPdenovo, on the grounds that it generally performs rather favorably compared to other assemblers, is a very general purpose assembler that can assemble small and large genomes, and it has favorable performance statistics. The latter is means simply that we want it to be able to run within the course of a morning session, even on a larger genome, such as human chromosome 14. Further information on SOAPdenovo:
http://soap.genomics.org.cn/soapdenovo.html

Evaluation of assemblies

The GAGE-paper (Salzberg et al. 2011) describes a number of detailed evaluation steps. Several of these evaluations use corrected assemblies. For instance, you can get a fantastic N50, but if the contigs are erroneously concatenated to very long scaffolds, this will create an unduly bias in some of the important statistics, such as N50. In the paper these corrections are done by first comparing to the published genome assemblies, then chopping up the scaffolds and contigs where there are errors. The analysis tools that can achieve all of these corrections and evaluations can be found at the GAGE website, and can also be found in the your GAGE_WageningenEU_repo directory ("gage-paper-validation"). Because of the complexity of these evaluations and – for human chromosome 14 – the long runtime of the scripts, we decided to stick to a simplified validation given the very short duration of this course practical.

Basic evaluations of the assemblies are done through a simple Perl script to calculate N50 and a few other parameters, on uncorrected assemblies. Furthermore, we are using MUMmer to align and evaluate the assemblies you've created to a published reference sequence. Besides MUMmer there are other tools that allow you to do similar things, such as MAUVE (http://asap.ahabs.wisc.edu/software/mauve/). We've chosen MUMmer based on the fact that it is widely used, and was also the tool of choice in the GAGE website and paper.
Further information on MUMmer:
http://mummer.sourceforge.net

For human chromosome 14 we will attempt to gain some insight from the local repeat structure on the assembly statistics, using an experimental script.

Two perl scripts have been emailed to you ('calculateN50.pl', and 'intersect_matches_and_repeats.pl'). The easiest way to work with these scripts is to copy them to the relevant working directories (~/GAGE_WageningenEU_repo/HChr14 and ~/GAGE_WageningenEU_repo/Staph).

Alternatively, create a folder 'bin', in your home directory (/home/ubuntu):

```
mkdir bin
```

Copy the scripts to this location. Make sure to make the scripts executable:

```
chmod +x scriptname
```

Finally, add the following line to your .bashrc:

```
export PATH=$PATH:/home/ubuntu/bin/
```

Assembly of *Staphylococcus aureus*

We are going to use the SOAPdenovo assembler to create an assembly. We are going to use two different sequence libraries for that. Navigate, from your home directory (/home/ubuntu), to the GAGE_WageningenEU_repo/Staph folder.

Remember you can navigate between folders on the command-line, for instance by doing:

```
cd GAGE_WageningenEU_repo/Staph
```

Navigating down in directory structure:

```
cd ..
```

Listing what is in a directory (sorted by newest files first):

```
ls –lth
```

Study the 'soapdenovo.config' file for location and make sure that you understand how the data located on your Virtual Machine relates to the data reported on the GAGE website (http://gage.cbcb.umd.edu/data/index.html)

The 'soapdenovo.config' file is a simple text file. You can see what is in a text file by doing 'cat' (this will list everything in the file). If the file is very big, you can list the first lines in a file using the 'head' command, or you can scroll through the file for instance by using 'more'.

Q: Which libraries are provided, and how do they differ from each other?

Since the k-mer based corrections by Quake are quite computation time and memory intensive, we have chosen to provide the corrected dataset instead of the original, raw data. This is the data you will use for the assembly.

Create assembly, in three steps:

The first step is creating the graph:

```
SOAPdenovo31mer pregraph -K 31 -s soapdenovo.config
-p 4 -o asm
```

One of the files that you have generated this way is: “test.kmerFreq”.

You can investigate the frequency distribution of K-mers by analyzing this file. For instance, in a separate terminal, open R (make sure you are starting R while in ‘Staph’ working directory). So, do:

```
R
```

You will now be on a different console. The ‘$’ prompt has been changed to ‘>’. You are now working in R.

Then do the following:

```
x <- read.table("asm.kmerFreq", header=F)
png("asm.kmerFreq.png",width=500, height=500)
plot(row.names(x),x$V1, xlim=c(1,50),
ylim=c(0,300000), col = 'dark red',xla = 'Depth',
ylab='Frequency')
dev.off()
q()
```

You will now have a file in the working directory, asm.kmerFreq.png that you can either display directly on your screen by doing:

```
display asm.kmerFreq.png
```

Alternatively, you can transfer the file to your local computer and display it there like a normal png file.

Q: Describe what you observe in the plot

The next command then assembles the contigs:

```
SOAPdenovo31mer contig -g asm
```

You can evaluate some basic stats on your newly assembled contigs:

```
cat asm.contig | perl calcN50.pl -m 200
```

Q: What is the largest contig? What are the N10, N50, N75, and N90 values?

Finally, remap the reads to the contigs, and do the scaffolding:

```
SOAPdenovo31mer map -s soapdenovo.config -g asm
SOAPdenovo31mer scaff -g asm
```

Now calculate some basic stats for the scaffolds:

```
cat asm.scafSeq | perl calcN50.pl -m 200
```

Q: What is the largest scaffold? What are the N10, N50, N75, and N90 values?

Finally, SOAPdenovo provides an add-on tool that can do gapclosing.

```
GapCloser -b soapdenovo.config -a asm.scafSeq -o
asm.2scafSeq -t 4 -p 31
```

Now scoll through the first scaffold of both the unclosed and closed sequences.

Q: How many stretches with Ns do you observe in the original scaffold sequence? And how many in the closed? How does this compare to the file "asm.2scafSeq.fill"?

Evalutation and visualization of the assembly using MUMmer

First we need to make a temporary file that contains the 'loose parts' , after GapClosing, from the scaffolds.

```
java -cp ~/GAGE_WageningenEU_repo/gage-paper-
validation/. SplitFastaByLetter asm.2scafSeq N
>tmp_scf.fasta
```

Then, MUMmer matches the published references seqeuence (St_aur_USA300_ref.fa) against the assembly that you have created yourself:

```
nucmer --maxmatch -p asm.2scafSeq -l 30
St_aur_USA300_ref.fa tmp_scf.fasta
```

Output files from nucmer need to be filtered and formatted further:

```
delta-filter -q asm.2scafSeq.delta
>asm.2scafSeq.fdelta

show-tiling asm.2scafSeq.fdelta >asm.2scafSeq.tiling

show-coords -rcl asm.2scafSeq.fdelta
>asm.2scafSeq.coords
```

You can subsequently visualize your assembly against the published reference. (Make sure that you have X-forwarding 'on').

```
mummerplot asm.2scafSeq.fdelta
```

Alternatively, you can have the file asm.dot.png file made, which you can then transfer to your local environment for viewing:

```
mummerplot -p asm.dot --png asm.2scafSeq.fdelta
```

Q: Which elements in this plot correspond to the scaffolds? And which to the remaining contigs? Why are some of the lines in opposite direction?

```
mummerplot asm.2scafSeq.tiling
```

Or, alternatively, the file asm.pip.png can be made for viewing on your local computer:

```
mummerplot -p asm.pip --png asm.2scafSeq.tiling
```

Q: How would you evaluate the overall coverage compared to the reference assembly? Are all regions covered?

If you want to find details on specific regions, on which scaffolds or contigs map exactly where to the reference genome, investigate the '.coords' file.

Effect of k-mer size on assembly statistics

There is usually no particular reason to choose a certain k-mer size, although one can devise rules of thumb based on the sequence length and technology, coverage, and complexity of the genome. For small genomes, a 'parameter-sweep' can be useful. You simply investigate a wide range of k-mer values and evaluate them on several critical parameters, such as N50. Simple shell scripts can help you to automate such a parameter sweep.

You can, for instance, have all the assemblies with odd k-mer values between 19 and 31 done in minutes using the following shell-one-liner. Note that the 'time' command merely keeps track of how much time it takes to executes the actual command(s) that follows it. Note that we do will use the 'SOAPdenovo all' option, which is equivalent to the three separate assembly steps you did in the previous section.

```
time for i in `seq 19 2 31`; do SOAPdenovo31mer all
-K $i -p 4 -s soapdenovo.config -o asm$i; done
```

You can then evaluate contig and scaffold parameters, again with shell-one-liners:

```
for i in `seq 19 2 31`; do NUMCONTIG=`cat
asm$i.contig | grep '>' | wc -l`; echo 'for k-mer size
'$i' there are '$NUMCONTIG' contigs'; cat asm$i.contig |
perl calcN50.pl -m 200 ; echo '' ; done;
```

```
for i in `seq 19 2 31`; do NUMSCAF=`cat asm$i.scaf |
grep scaf | wc -l`; echo 'for k-mer size '$i' there are
'$NUMSCAF' scaffolds'; cat asm$i.scafSeq | perl
calcN50.pl -m 200 ; echo '' ; done;
```

Q: How are the assemblies affected by k-mer size?

Optional: Effect of assigning the wrong direction for scaffolding

It is critical to assign the correct direction to the longer insert libraries. Since these libraries are often derived from circularized fragments, the 'reverse seq' flag needs to be set to '1'. You can investigate the consequence of having the 'reverse_seq' flag wrong by changing for the second library (short jump):
reverse_seq=1
to:
reverse_seq=0

Q: How does assigning the wrong 'reverse_seq' flag affect the scaffolding statistics?

Assembly of Human Chromosome 14

Human chromosome 14 provides a far larger challenge because of projected size (~107Mbp), total volume of data to be analysed, and complexity of the genome. To speed up the assembly, we will use the 'all' option of SOAPdenovo, so we can skip typing in the three different steps in SOAPdenovo. We will also skip the GapClosing, due to time constraints. We will use the k-mer size of 47, which was also used in the GAGE website.

Assembly:

```
time SOAPdenovo63mer all -K 47 -p 4 -s
soapdenovo.config -o asm47
```
(takes about 25 minutes)

Then, evaluate the basic stats of the contigs:
```
cat asm47.contig | perl calcN50.pl -m 200
```

And of the scaffolds:
```
cat asm47.scafSeq | perl calcN50.pl -m 200
```

Q: What are the contig and scaffold N50? How does this compare to the Staphylococcus aureus assembly you made before?

MUMMer evaluation:

First we need to make a temporary file that contains the remaining 'loose parts' , after GapClosing, from the scaffolds. Essentially, this means that the scaffolds are broken up where there are stretches of Ns.

```
java -cp  ~/GAGE_WageningenEU_repo/gage-paper-
validation/. SplitFastaByLetter asm47.scafSeq N
>tmp_scf.fasta
```

```
time nucmer --maxmatch -p asm47.scafSeq -l 30
chr14.fa tmp_scf.fasta
```
(takes ~25 minutes)

```
delta-filter -q asm47.scafSeq.delta
>asm47.scafSeq.filter

show-tiling asm47.scafSeq.filter
>asm47.scafSeq.tiling

show-coords -rcl asm47.scafSeq.filter
>asm47.scafSeq.coords
```

Then, again visualize the result by:

```
mummerplot asm47.scafSeq.tiling
```

Alternatively, you can create a file asm47.pip.png file made, which you can then transfer to your local environment for viewing:

```
mummerplot -p asm47.pip --png asm47.scafSeq.tiling
```

And:

```
mummerplot asm47.scafSeq.filter
```

In this plot, zoom in to any one of the scaffolds. You can do this by right-clicking to create a selection-box, which you can then increase in size by moving your mouse. Once you are satisfied by the size of the box, left-click to finish and allow some time for the zoomed image to render on your screen.

Alternatively, create the file asm47.dot.png file made, which you can then transfer to your local environment for viewing:

```
mummerplot -p asm47.dot --png asm47.scafSeq.fdelta
```

Q: Why does the assembly look so much more fragmented than the *Staphylococcus aureus* one, even though the N50 values are comparably good?

Investigate the asm47.scafSeq.coords file for further details on the comparison between your assembly and the published reference.

One of the things you will note in the mummerplots, and from the '.coords' file, is that the comparisons start at exactly bp 19 million. The reason is that, Chr14 being more or less acrocentric, the first part of the chromosome has no sequence in the reference assembly. The highly repetitive and heterochromatic nature of the chromosome near the centromeric/telomeric part makes it very difficult to sequence.

Check out the karyotype of human chromosome 14 yourself:
http://www.ensembl.org/Homo_sapiens/Location/Genome

One other thing you will notice, is, that although your assembly contains a lot of quite long contigs and scaffolds, if you visualize it against the entire published contiguous sequence of chromosome 14, it looks extremely fragmented and messy. This is the current reality for 'De Novo assembly'. Do not expect a finished chromosome, but rather a large bag of contigs and scaffolds. The big challenges for De Novo assembly currently are how to turn these into meaningful larger structures, preferably consistent with entire linkage groups.

Investigate effect of local repeat structure

In general, you can expect repeats, from small (SINEs such as the Alu sequences) to large (LINEs, such as the ubiquitous L1s) to create problems in genome assembly. As the final part of the exercise, we are going to evaluate contig assembly statistics as a function of the local repeat structure in the genome. We will focus on the Alu sequences (of which ~1 million occur in the human genome, and are ~300bp in length). If you wish, you can do the same for other types of repeats.

For this exercise we will use an experimental Perl script, that will intersect the '.coords' file with the repeat annotations (which can be found in the file: repeats_chr14.bed; data taken from UCSC web browser). The Perl script also invokes R to make the correlation analysis and do a simple plot of the relationship of number of Alu sequences and the highest length of contig sizes found, in bins of 100kbp.

```
cat asm47.scafSeq.coords | perl
intersect_matches_and_repeats.pl  -f repeats_chr14.bed -q
Alu
```

The script will generate a figure of the relationship between local Alu content, and assembly success, as measured by the length of the largest local contig. The Rin.Rout file, in addition, will show the correlation coefficient between these two parameters.

Q: how are assembly success and local Alu content related?