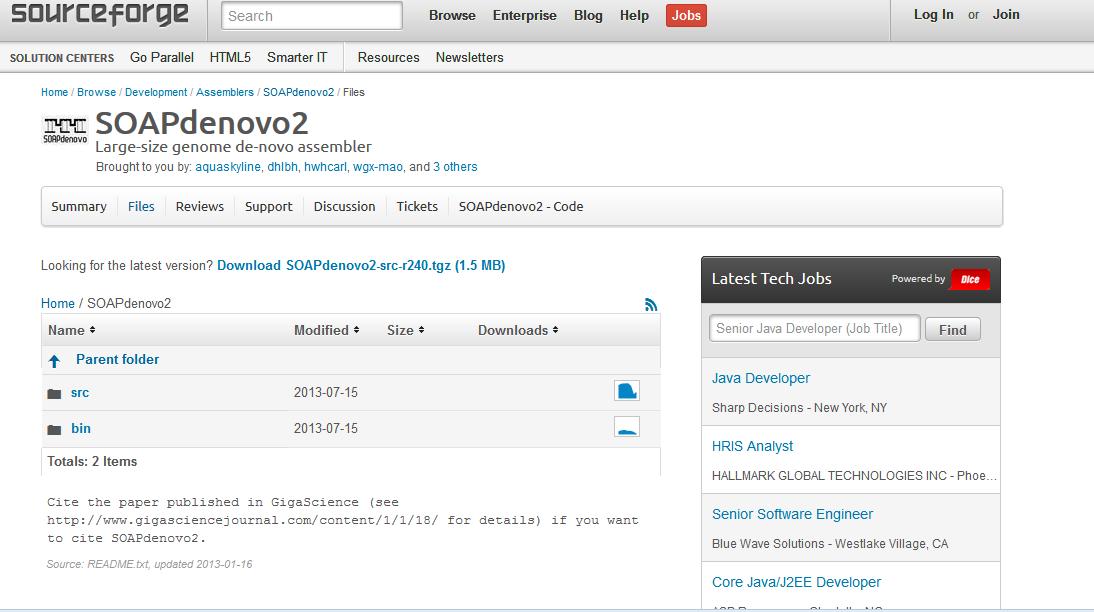
SOAPdenovo uitleg/userguide  
  
SOAP is an assembler tool which uses debruijn graphs and runs on all Linux distributions, it can be downloaded from <http://sourceforge.net/projects/soapdenovo2/files/SOAPdenovo2/>,SOAP can be found in two different flavours a bin and a src version. The bin version only has to be extracted and is usable right after, the src version instead has to be compiled for further use.



Figuur 1

After one of the versions has been downloaded it can be extracted in the terminal with the command( in the correct folder containing the appropriate files like, cd /Downloads/SOAP\*) "tar -zxf ${destination folder} SOAPdenovo2-src-r240.tgz" for both the bin and the src versions. The src version needs to be compiled before it can be used, to compile the src version use the command "make" in the extracted folder. After that "make install" has to be used to the install executable.

When SOAP is compiled or extracted it can be started with ./SOAPdenovo-63mer(or 127mer)but to actually run SOAP with a dataset a configfile is needed, an template with commentary can be found in the bijlage. The major difference between SOAPdenovo-63mer and 127mer is that 127mer has double memory consumption and also accepts kmer sizes greater then 63. For usage with kmers below 63 its adviceble to use always 63mer because even with a kmer setting below 63, 127mer uses twice the amount of memory.

The config file from SOAP gives the user a lot of options to adjust the assembly, like read length ETC AANVULLEN. But the most important thing in the config file are the paths to the fasta or fastq files.

A full list with all options can be found below;

#maximal read length

max\_rd\_len=100

[LIB]

#average insert size

avg\_ins=200

#if sequence needs to be reversed

reverse\_seq=0

#in which part(s) the reads are used

asm\_flags=3

#use only first 100 bps of each read

rd\_len\_cutoff=100

#in which order the reads are used while scaffolding

rank=1

# cutoff of pair number for a reliable connection (at least 3 for short insert size)

pair\_num\_cutoff=3

#minimum aligned length to contigs for a reliable read location (at least 32 for short insert size)

map\_len=32

Like stated before one of the most important things in the config file are the paths to the fasta or fastq files, these paths are different for all sorts of sequence files like fasta and fastq and paired and single reads. Example for a pair of paired end files the path should be; (example states fa en fq files but the files don’t need to be extracted from their gz files)

f1=/path/folderwithsequences/fasta1\_read\_1.fa

f2=/path/folderwithsequences/fasta1\_read\_2.fa  
When the data consist out of fastq files the paths should be;

q1=/path/folderwithsequences/fastq1\_read\_1.fq

q2=/path/folderwithsequences/fastq1\_read\_2.fq

For single end reads the paths are almost equal for a fastq file;  
q=/path/folderwithsequences/fastq1\_read\_single.fq

For a fasta file;

f=/path/folderwithsequences/fasta1\_read\_single.fa

When the paths are sorted it is important to remove the unused paths in the config file because SOAP won’t run with them.

OPTIES  
  
When the options are adjusted on the config file SOAP can be run with the command; ./SOAPdenovo-63mer all -s configfilename.config -K 63 -R -o PREFIX>ass.log 2>ass.err for 63mer and ./SOAPdenovo-127mer all -s configfilename.config -K 63 -R -o PREFIX>ass.log 2>ass.err for 127mer, these commandline options require a few required parameters but also has a few optional parameters. The parameters which are required are:

-s <string> configFile: the config file of solexa reads  
-o <string> outputGraph: prefix of output graph file name  
 -K <int> kmer(min 13, max 63/127): kmer size, [23]  
-o <string> outputGraph: prefix of output graph file name  
-R (optional) resolve repeats by reads, [NO] not required but higly recommanded

Optional parameters are:

-p <int> n\_cpu: number of cpu for use, [8]  
 -a <int> initMemoryAssumption: memory assumption initialized to avoid further reallocation, unit G, [0]

-d <int> KmerFreqCutoff: kmers with frequency no larger than KmerFreqCutoff will be deleted, [0]  
-D <int> EdgeCovCutoff: edges with coverage no larger than EdgeCovCutoff will be deleted, [1]  
 -M <int> mergeLevel(min 0, max 3): the strength of merging similar sequences during contiging, [1]  
-m <int> max k when using multi kmer  
-e <int> weight to filter arc when linearize two edges(default 0)  
-r (optional) keep available read(\*.read)  
 -E (optional) merge clean bubble before iterate  
 -f (optional) output gap related reads in map step for using SRkgf to fill gap, [NO]  
 -k <int> kmer\_R2C(min 13, max 63): kmer size used for mapping read to contig, [K]  
 -F (optional) fill gaps in scaffold, [NO]  
-u (optional) un-mask contigs with high/low coverage before scaffolding, [mask]  
 -w (optional) keep contigs weakly connected to other contigs in scaffold, [NO]  
 -G <int> gapLenDiff: allowed length difference between estimated and filled gap, [50]  
 -L <int> minContigLen: shortest contig for scaffolding, [K+2]  
 -c <float> minContigCvg: minimum contig coverage (c\*avgCvg), contigs shorter than 100bp with coverage smaller   
 than c\*avgCvg will be masked before scaffolding unless -u is set, [0.1]  
 -C <float> maxContigCvg: maximum contig coverage (C\*avgCvg), contigs with coverage larger than C\*avgCvg or contigs shorter than 100bp with coverage larger than 0.8\*C\*avgCvg will be masked before scaffolding unless -u is set, [2]  
-b <float> insertSizeUpperBound: (b\*avg\_ins) will be used as upper bound of insert size for large insert size  
 ( > 1000) when handling pair-end connections between contigs if b is set to larger than 1, [1.5]  
-B <float> bubbleCoverage: remove contig with lower cvoerage in bubble structure if both contigs' coverage are smaller than bubbleCoverage\*avgCvg, [0.6]  
-N <int> genomeSize: genome size for statistics, [0]  
-V (optional) output visualization information of assembly, this option creates an .asm file which can be used in Amos Hawkeye viewer.

After the command is written to the needs of the user it can be run with the press of enter, now if everything is correct SOAP will start with the assembly. The terminaloutput with runtime, and results like n50 etc.. wil be written in the logfile which can be found in the SOAP folder this is very usefull to check the quality of the assembly The runtime depends greatly on the amount of sequences in the file(s), the runtime can last hours or even days. When SOAP is finished it will write a few output files.

https://wiki.nbic.nl/index.php/Raw\_results\_of\_NGS\_de\_novo\_assembly

If there is no refence sequence to allign your sequences to, you will have to assemble the genome de novo. There are several software packages to do this. A review of the performance of the various packages can be found [here](https://wiki.nbic.nl/index.php/Raw_results_of_NGS_de_novo_assembly). There have also been a number of recent benchmarkering studies that compare the performance of available packages, e.g. [Assemblathon\_1](http://genome.cshlp.org/content/21/12/2224) and [GAGE](http://genome.cshlp.org/content/early/2012/01/05/gr.131383.111).  
  
De novo assembly is a very memory-intensive process. For anything exceeding a small bacterial it requires access to a sever with a substantial amount of RAM. [SOAPdenovo](http://soap.genomics.org.cn/soapdenovo.html) and [ABySS](http://www.bcgsc.ca/platform/bioinfo/software/abyss) are more workable than for example Velvet. These programs have the advantage that they can be run over multiple cores, which can speed up parts of the process. People are apparently also working on [more efficient ways](http://genome.cshlp.org/content/early/2011/12/07/gr.126953.111) to do de novo assembly.

Meanwhile, I've had good results using SOAPdenovo. This assembler has few options to tweak. The most important one is the choice of k-mer length, or the word length used in the assembly process. There are no set rules for this and the choice is usually based on trial-and-error. In other words, you'll have to run the assembly a number of times with different k-mers (those of us into scipting, might write a little bash script to do this (or get someone else to write it for you). To run SOAPdenovo, you have to prepare a config file (see the example in the readme file) and then run the following:

$ SOAPdenovo all -s [config\_file] -o [output\_prefix] -K [k-mer size] -p [number\_of\_cores\_to\_use]

The summary statistics (max contig length, n50, etc.) are written to the terminal. It is useful to capture this information in a separate file like this:

$ SOAPdenovo all -s [config\_file] -o [output\_prefix] -K [k-mer size] -p [number\_of\_cores\_to\_use] > [log\_file]  
  
The output of the program consists of a number of files, of which the one ending .contig contains your contigs and .scafSeq the scaffolds.

bin versie

config file:

#iets laten draaien

./SOAPdenovo-63mer all -s configfilename.config -K 63 -R -o graph\_prefix1 > ass.log 2 > ass.err

-o is een prefix voor de bestanden

of 127mer, voor testen zie config.test in de soapdenovo map

q1=~/Downloads/C0A7AACXX\_101851-02\_TGACCA\_L001\_R2.fastq.gz

http://assemblathon.org/

Bijlage configfile (save as yourname.config)

#maximal read length

max\_rd\_len=100

[LIB]

#average insert size

avg\_ins=200

#if sequence needs to be reversed

reverse\_seq=0

#in which part(s) the reads are used

asm\_flags=3

#use only first 100 bps of each read

rd\_len\_cutoff=100

#in which order the reads are used while scaffolding

rank=1

# cutoff of pair number for a reliable connection (at least 3 for short insert size)

pair\_num\_cutoff=3

#minimum aligned length to contigs for a reliable read location (at least 32 for short insert size)

map\_len=32

#weghalen wat niet wordt gebruikt

#maar vier files totaal of?

#a pair of fastq file, read 1 file should always be followed by read 2 file

q1=/path/\*\*LIBNAMEA\*\*/fastq1\_read\_1.fq

q2=/path/\*\*LIBNAMEA\*\*/fastq1\_read\_2.fq

#another pair of fastq file, read 1 file should always be followed by read 2 file

q1=/path/\*\*LIBNAMEA\*\*/fastq2\_read\_1.fq

q2=/path/\*\*LIBNAMEA\*\*/fastq2\_read\_2.fq

#a pair of fasta file, read 1 file should always be followed by read 2 file

f1=/path/\*\*LIBNAMEA\*\*/fasta1\_read\_1.fa

f2=/path/\*\*LIBNAMEA\*\*/fasta1\_read\_2.fa

#another pair of fasta file, read 1 file should always be followed by read 2 file

f1=/path/\*\*LIBNAMEA\*\*/fasta2\_read\_1.fa

f2=/path/\*\*LIBNAMEA\*\*/fasta2\_read\_2.fa

#fastq file for single reads

q=/path/\*\*LIBNAMEA\*\*/fastq1\_read\_single.fq

#another fastq file for single reads

q=/path/\*\*LIBNAMEA\*\*/fastq2\_read\_single.fq

#fasta file for single reads

f=/path/\*\*LIBNAMEA\*\*/fasta1\_read\_single.fa

#another fasta file for single reads

f=/path/\*\*LIBNAMEA\*\*/fasta2\_read\_single.fa

#a single fasta file for paired reads

p=/path/\*\*LIBNAMEA\*\*/pairs1\_in\_one\_file.fa

#another single fasta file for paired reads

p=/path/\*\*LIBNAMEA\*\*/pairs2\_in\_one\_file.fa

#bam file for single or paired reads, reads 1 in paired reads file should always be followed by reads 2

# NOTE: If a read in bam file fails platform/vendor quality checks(the flag field 0x0200 is set), itself and it's paired

read would be ignored.

b=/path/\*\*LIBNAMEA\*\*/reads1\_in\_file.bam

#another bam file for single or paired reads

b=/path/\*\*LIBNAMEA\*\*/reads2\_in\_file.bam

[LIB]

avg\_ins=2000

reverse\_seq=1

asm\_flags=2

rank=2

# cutoff of pair number for a reliable connection (at least 5 for large insert size)

pair\_num\_cutoff=5

#minimum aligned length to contigs for a reliable read location (at least 35 for large insert size)

map\_len=35

#weghalen wat niet wordt gebruikt

q1=/path/\*\*LIBNAMEB\*\*/fastq\_read\_1.fq

q2=/path/\*\*LIBNAMEB\*\*/fastq\_read\_2.fq

f1=/path/\*\*LIBNAMEA\*\*/fasta\_read\_1.fa

f2=/path/\*\*LIBNAMEA\*\*/fasta\_read\_2.fa

p=/path/\*\*LIBNAMEA\*\*/pairs\_in\_one\_file.fa

b=/path/\*\*LIBNAMEA\*\*/reads\_in\_file.bam