Méthodes Yeast:

Quality control of sequences

The high throughput sequencing method isn’t infaillble, so the data will have contaminants, badly read sequences due to the intensity of the fluorescent signal, or the quality of the reagents that have a decaying quality with the number of sequencing cycles over time. (10.4172/jpb.1000420 )These errors might add a lot of false signals, useless extrawork, and complicate the data analysis, therefore we need to get rid of them.

In order to check for the quality of the data, we use the tool : ‘fastqc’ which will help us visualize our fasta file given by the sequencer.

(<http://hannonlab.cshl.edu/fastx_toolkit/>) on fastqc

Trimming and bad quality removal

After the quality control check, we used the tool ‘trimmomatic’ with the given parameters of MINLEN : 130 to trim down the bad quality ends of the reads, keeping at least 130bp of the trimmed read, and the parameter SLIDINGWINDOW:4:15, thus removing the reads that have an average base pair quality score lower than 15.

(http://www.usadellab.org/cms/?page=trimmomatic)

The next step was checking if the quality of the data has improved after the trimming process, by using again fastqc, on the trimmomatic fastq file output.

Sequence alignment against reference

Since the fasta gives no information about the sequences position in the yeast genome, we had to align all of the reads again against fasta file of a known yeast genome, or most likely a consensus of a yeast genome, containing the positional information.

However, in order to do that, we first had to index the reference fasta using the ‘bwa index’ tool, which is a way of giving a sort of ‘table of contents’ of our reference fasta file (in our case the R64-1-1.92.fa), that are used by the burrows-wheeler aligner algorithm. Subsequently, we used the ‘bwa mem’ tool in order to align our sequenced data against the reference. Then we have converted all the samfiles containing our data (given as an output after the alignment), into bam files (binary format) in order to work with them.

<https://doi.org/10.1093/bioinformatics/btp698> (On bwa)

Variant calling and annotation

We have done this step using only samtools mpileup, that took our reference file, and the aligned bamfiles as an input, and gave us the binary format of the variant calling format files. Then the script piped the output of the samtools command into ‘bcftools’ in order it into vcf.gz files. This method was prefered considering the fact that our genome is a haploid yeast genome, and it doesn’t need a complicated algorithm as used by the GATK pipeline.

https://github.com/samtools/hts-specs (samtools info)

<http://www.htslib.org/doc/samtools-1.2.html> (also this is for samtools)

The ‘tabix’ tool was used on the vcf files, in order to index them properly.

<https://doi.org/10.1093/bioinformatics/btq671> (on tabix)

In order to annotate the variants given by the vcf files, we had to use ‘SNPeff’ tool on the vcf files that were merged together with all their indexes, and we also kept only the variants that were found in less than all 4 strains that we had to analyze, since we had to filter through all the variants that were different from the reference. The variants interesting to us, are indeed the ones that are specific to one of the mutants, and that’s why we had to filter this way.

(<http://snpeff.sourceforge.net/download.html#download>) on SnpEff

The results were then visualized by either reading the vcf files in xcel, and by IGV.