Analysis of Exome Data

Disclaimer:
The methods described in this guideline describe the way I do exome sequencing analysis in my laboratory (of humans) and may not be suitable for every . Feel free to add, change, remove parts of it (in a sensible manner). As I am not heavily experienced neither in bioinformatics in general nor in next-generation sequencing, these recommendations should be handled with care...
Furthermore I only deal with Illumina data, some people more experienced using data generated by other platforms could contribute their challenges.
And finally: I use Linux operating system and make some use of the command line, so most of the parts below may not work properly on different distributions and will definitely not work in Windows...

Exome sequencing analysis can be divided in three major steps:

- Base-calling and image analysis
- Alignment
- SNP-calling

While the first step is mostly done within the laboratories of the sequence service provider, I will only discuss the latter two. That sounds easy but each step requires specific preparations, adjustments and knowledge of possible mistakes and artifacts which may arise during the sequencing process. I will discuss the way I do exome-sequencing analysis in my laboratory and I hope that this is to some use for somebody. As we only started doing exome analysis last year and we have done only some 15 exomes so far, there might still be things overlooked or wrong. Please tell me if that is the case...

1. FastQ Files

Usually data from a sequence service provider comes as FastQ Files, which looks like the following example for Illumina data:

```
@EAS100R:136:FC706VJ:2:2104:15343:197393 1:Y:18:ATCAG
GATTTGGGGTTCAAAGCAGTATCGATCAAATAGTAAATCCATTTGTTCAACTCACAGTTT
+
!''*((((***+))%%%++)(%%%%).1***-+*''))**55CCF>>>>>>CCCCCCC65
```

These four lines represent a single read. The first line (always starting with '@') specifies the name of the read. In case of Illumina sequences it also contains information about the position on the flow-cell in the following manner:

```
@<instrument name>:<run-id>:<flow-cell-id>:<flowcell lane>:<tile within the
flow-cell lane>:<x-coordinate of the detected cluster>:<y-Position of the
detected cluster> <member of a pair>:<Y if the read is filtered, N otherwise>:<0
when control bits are on, even number otherwise>:<multiplex index>
```

The second line is the actual sequence of the read. A,C,G,T and N characters are allowed in this line. N is used if the real base couldn't be determined.

The third line contains either only the '+' sign or it may contain the read name starting after the '@' sign.

Finally the last line is the ASCII encoding of the quality of the base exactly two lines above. There are several different ways of encoding the quality and many of them are allowed in the FastQ format. The latest and in sequencing service most common way is to output the quality is the so-called „Illumina1.8+“ format (which is actually the so-called Sanger format). The qualities are Phred Scaled which means the machine calculates the error probability of having called the base wrong, takes the decadic logarithmus and multiplies it by -10. So Q20 means a probability of 0.01 of having a base wrong, Q30 a probability of 0.001 and so on. The Illumina1.8+ encoding scheme encodes that Phred-score by adding 33 to the score. The resulting value is encoded in ASCII Code for saving disk space in a text file (the value 40 needs two bytes the ASCII correspondent “(“ only takes one byte).

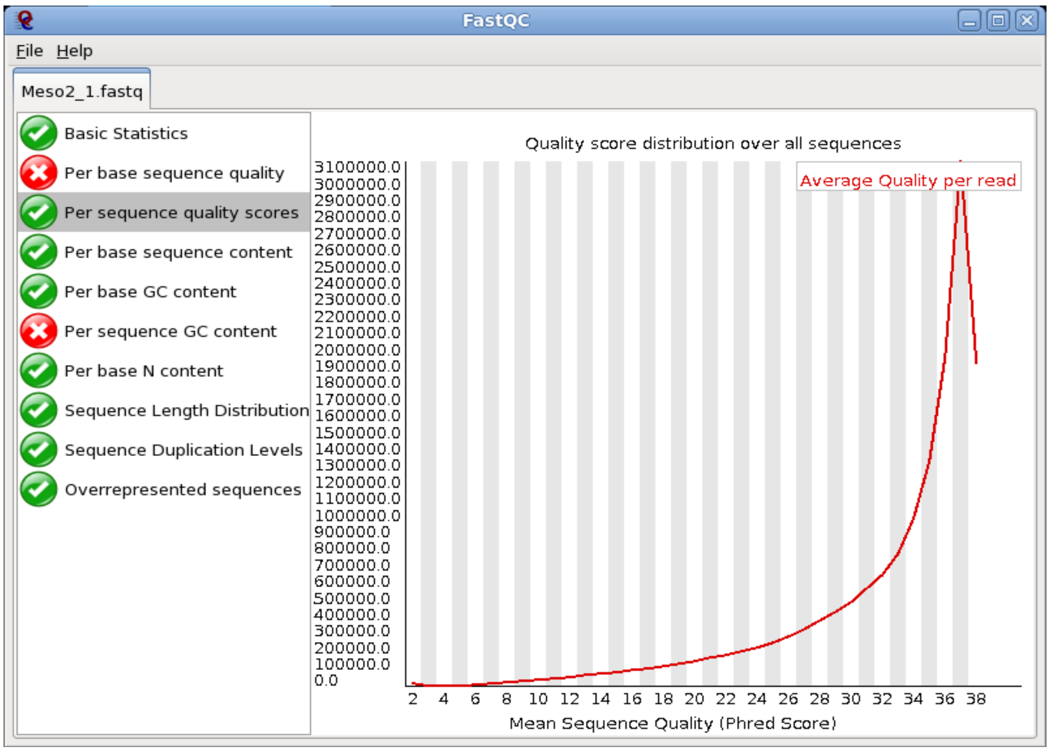
Earlier versions of FastQ File format (Illumina 1.5+) generated by Illumina added 64 to the Phred Score and encoded it from ASCII 66-126, however ASCII 66 had a different meaning: it was used to mark bases which should not be considered in downstream analyses.

Illumina version 1.3+ uses phred scores 0-62 encoded in ASCII 64-126.

Exome data is often generated using a paired end approach, which means sequences are retrieved from both ends of the same molecule. As the molecules are larger than twice the read length of an average Illumina experiment, these sequences do not overlap, but there is space in-between (which is referred to as „insert sequence“). This approach helps to detect PCR duplicates which we will discuss later-on.

Sequence data from paired end approaches come in two files: One stores all the forward sequences and the other stored the reverse sequences in the exact same order as the first file.

For inspection of the quality of the FastQ Files you could use the FastQC[1] package which is easy to use and outputs some nice graphics such as read-length plots, read-quality plots, sequence-duplication levels and many more. FastQC comes wit a start script which can be executed on every platform.


FastQC
File Help
Meso2_1.fastq
Basic Statistics
Per base sequence quality
Per sequence quality scores
Per base sequence content
Per base GC content
Per sequence GC content
Per base N content
Sequence Length Distribution
Sequence Duplication Levels
Overrepresented sequences
Quality score distribution over all sequences
Average Quality per read
Mean Sequence Quality (Phred Score)

Average quality per read plot generated by FastQC.

1 http://www.bioinformatics.bbsrc.ac.uk/projects/fastqc/

2. Alignment

2.1. Preparation of input files

For analyzing sequence data, alignment to a reference genome is necessary. For humans there are several options of retrieving the reference sequence: the most important being UCSC and ensembl.

The newest UCSC release of the human genome s called hg19 and can be accessed via the following link: http://hgdownload.cse.ucsc.edu/goldenPath/hg19/bigZips/chromFa.tar.gz

As the downstream applications need a single sorted FastA file (which is a text-based format with a header line starting with the '>' sign and the sequence in the following lines; multiple sequences can be merged in a single fasta file when they are separated by header lines) of the genome, we produce that by invoking the cat command on the Linux command line, which links all of the files to a single file (which I call hg19.fa).

At first unpack the .tar.gz file

```
tar -xzf chromFa.tar.gz
```

Then concatenate the single-chromosome files to a single genome reference file (make sure they are in the exact same order as stated below, GATK won't work otherwise):

```
cat chr1.fa chr2.fa chr3.fa chr4.fa chr5.fa chr6.fa chr7.fa chr8.fa chr9.fa
chr10.fa chr11.fa chr12.fa chr13.fa chr14.fa chr15.fa chr16.fa chr17.fa chr18.fa
chr19.fa chr20.fa chr21.fa chr22.fa chrX.fa chrY.fa chrM.fa > hg19.fa
```

For aligning the sequences to the human genome I use BWA[2]. It needs very few memory, works well with Illumina data and can be run using several threads. For using BWA, the reference genome needs to be indexed and transformed using an indexing program supplied via the BWA SourceForge page.

```
bwa index -a bwtsw -p hg19 hg19.fa
```

That may take a while. hg19 is the prefix for some files that will be generated by that command and this prefix will be used later-on in the alignment steps. The -a switch tells the indexing program to use the bwtsw algorithm for constructing the reference files, as the other algorithms won't work for a human-size genome. For a list of all the options type bwa index without arguments.

2.2. Actual Alignment

If this is done you could start aligning you fastq files to that by invoking BWA like this:

```
bwa aln -t 4 -f input.sai -I hg19 input.fastq
```

BWA uses many different options to adjust the alignment only a few are used in the statement above. The -t options tell BWA the number of threads to use (4 in this case), while the -I option tells BWA to use Illumina1.3+ qualities. Hg19 tells the program to use the file we generated one step earlier as a reference. For more information or more details on the BWA program see: http://bio-bwa.sourceforge.net/bwa.shtml or simply type bwa aln for a list of all the options.

If you got two files per sample be sure two align both files separately before doing the next step.

BWA outputs the alignments in sai format, however, as most downstream-analysis programs make use of the SAM (Sequence Alignment/Map format) file format[3] you need to transform the output using the bwa samse and sampe command, respectively.

For single end reads (i.e. One file per sample) do the following:

```
bwa samse -f out.sam  -r
```

2 http://bio-bwa.sourceforge.net/

3 See http://samtools.sourceforge.net/SAM1.pdf for a complete description of the file format

```
"@RQ\tID:<ID>\tLB:<LIBRARY_NAME>\tSM:<SAMPLE_NAME>\tPL:ILLUMINA" hg19 input.sai
input.fq
```

This looks a bit confusing but the -r Argument (the so-called RG-line) is necessary for SNP calling using the GATK package. This line will be put in the generated SAM file specifying the ID, the library name and the sample name of the sequences and the sequencing platform. As most service providers will hand out the data with one FastQ file per sample, I usually put the same name for the ID the LB and the SM field. For a sample called Exome1 that would in my case look like:

```
 „@RG\tID:Exome1\tLB:Exome1\tSM:Exome1\t:PL:ILLUMINA“
```

Again, hg19 means the prefix specified during the transformation of the reference file input.sai is the result of the alignment step and input.fq is the input FastQ file.

For paired-end data there is a separate command which looks like this:

```
bwa sampe -f out.sam  -r
"@RQ\tID:<ID>\tLB:<LIBRARY_NAME>\tSM:<SAMPLE_NAME>\tPL:ILLUMINA" hg19 input1.sai
input2.sai input1.fq input2.fq
```

Again the result is a SAM file which is needed for later analysis.

2.3. SAM to BAM conversion

The SAM file is the starting point for obtaining a much more powerful file type: the binary Alignment/Map format (short BAM). It compresses the SAM file and can be indexed, which means that only portions of the file can be accessed without the need to load the whole file. A usual exome BAM file takes about 4-10 Gb disk space while a SAM file needs 20-30 Gb.

For converting the SAM file to BAM we use picard[4]. Picard offers many options for manipulating or viewing SAM and BAM files. We will use the SortSam.jar program to sort the SAM file and save it as a BAM file:

```
java -Xmx4g -Djava.io.tmpdir=/tmp \
     -jar picard/SortSam.jar \
     SO=coordinate \
     INPUT=input.sam \
     OUTPUT=output.bam \
     VALIDATION_STRINGENCY=LENIENT \
     CREATE_INDEX=true
```

For having a better overview I put the arguments in separate lines and join them in a script by the “\” sign.

The file types are recognized by their endings, so be sure to have them named as “.sam” and “.bam” files. The SO argument specifies the sort order which is by coordinate in our case. By setting the validation stringency to lenient, picard ignores some validation errors which frequently occur at the alignment step. By setting the CREATE_INDEX argument to true we automatically create an index file for the generated bam file. This file got the same name as the bam file but with the additional extension “.bai” (so Exom1.bam got the index file Exom1.bam.bai).

2.4. Marking PCR duplicates

Due to inherent mistakes in the sequencing technology, some reads will be exact copies of each other. These reads are called PCR duplicates due to amplification biases in PCR. They share the same sequence and the same alignment position and could cause trouble during SNP calling as

4 http://picard.sourceforge.net/index.shtml

possibly some allele is overrepresented due to amplification biases. This procedure works better for paired-end approaches as the probability of having both sides of two molecules aligned on the same position and the sequence being the same by random is very low (but not impossible).

Again picard is used to mark possible PCR duplicates in the BAM file by setting a specific flag.

```
java -Xmx4g -Djava.io.tmpdir=/tmp \
     -jar picard/MarkDuplicates.jar \
     INPUT=input.bam \
     OUTPUT=input.marked.bam \
     METRICS_FILE=metrics \
     VALIDATION_STRINGENCY=LENIENT
```

Again the validation stringency is set to lenient but no index is created as we will perform more changes on the output bam file.

2.5. *Local realignment around indels*

Indels within reads often lead to false positive SNPs at the end of sequence reads. To prevent this artifact, local realignment around indels is done using local realignment tools from the Genome Analysis Tool Kit (short: GATK)[5]. This is done in two steps: the first step creates a table of possible indels and the second step realigns reads around those targets:

Step1:

```
java -Xmx4g -jar GenomeAnalysisTK.jar \
     -T RealignerTargetCreator  \
     -R hg19.fa \
     -o input.bam.list \
     -I input.marked.bam
```

This step puts the table in the file in input.bam.list . When this is finished we can start the realigning step using the statements below:

```
java -Xmx4g -Djava.io.tmpdir=/tmp \
     -jar GenomeAnalysisTK.jar \
     -I input.marked.bam \
     -R hg19.fa \
     -T IndelRealigner \
     -targetIntervals input.bam.list \
     -o input.marked.realigned.bam
```

When using paired end data, the mate information must be fixed, as alignments may change during the realignment process. Picard offers a utility to do that for us:

```
java -Djava.io.tmpdir=/tmp/flx-auswerter \
  -jar picard/FixMateInformation.jar \
  INPUT=input.marked.realigned.bam \
```

5 http://www.broadinstitute.org/gsa/wiki/index.php/Downloading_the_GATK

```
  OUTPUT=input_bam.marked.realigned.fixed.bam \
  SO=coordinate \
  VALIDATION_STRINGENCY=LENIENT \
  CREATE_INDEX=true
```

2.6. Quality score recalibration

That's still not all. Quality data generated from the sequencer isn't always very accurate and for obtaining good SNP calls (which rely on base quality scores), recalibration of these scores is necessary[6]. Again this is done in two steps: the CountCovariates step and the TableRecalibration steps. Both can be run from the GATK package:

1) Count covariates:

```
java -Xmx4g -jar GenomeAnalysisTK.jar \
     -l INFO \
     -R hg19.fa \
     --DBSNP dbsnp132.txt \
     -I input.marked.realigned.fixed.bam \
     -T CountCovariates \
       -cov ReadGroupCovariate \
       -cov QualityScoreCovariate \
       -cov CycleCovariate \
       -cov DinucCovariate \
       -recalFile input.recal_data.csv
```

This step creates a .csv file which is needed for the next step and requires a dbSNP file, which can be downloaded at the UCSC Genome browser homepage[7]. DbSNP132 is the most novel one which can be downloaded from the UCSC browser, but dbSNP is updated regularly, so newer versions will be available in the future. Download the dbsnp132.txt.gz file and unzip it using gunzip (that's just an example).

2) Table recalibration:

```
java -Xmx4g -jar GenomeAnalysisTK.jar \
  -l INFO \
  -R hg19.fa \
  -I input.marked.realigned.fixed.bam \
  -T TableRecalibration \
   --out input.marked.realigned.fixed.recal.bam \
   -recalFile input.recal_data.csv
```

Believe it or not, but now we are ready for getting some SNP calls...

6 See http://www.broadinstitute.org/files/shared/mpg/nextgen2010/nextgen_poplin.pdf as well
7 http://hgdownload.cse.ucsc.edu/goldenPath/hg19/database/

3. SNP calling

3.1. *Produce raw SNP calls*

SNP calling is done using the GATK UnifiedGenotyper program. It calls SNPs and short indels at the same time and gives a well annotated VCF file as output.

```
java -Xmx4g -jar GenomeAnalysisTK.jar \
     -glm BOTH \
     -R hg19.fa \
     -T UnifiedGenotyper \
     -I input.marked.realigned.fixed.recal.bam \
     -D dbsnp132.txt \
     -o snps.vcf \
     -metrics snps.metrics \
     -stand_call_conf 50.0 \
     -stand_emit_conf 10.0 \
     -dcov 1000 \
     -A DepthOfCoverage \
     -A AlleleBalance \
     -L target_intervals.bed
```

The “glm both” argument enables SNP and Indel calling. The parameters above work well for my situation, but many changes can be done in that step. For a detailed explanation see:
http://www.broadinstitute.org/gsa/wiki/index.php/Unified_genotyper

The metrics file specified by the metrics command contains statistics for the SNP calling step and the snps.vcf file contains the raw snp calls.

The -L command contains a list of the targeted intervals we wanted to sequence. This is necessary as only about 60-70% of all the reads will end up in exonic regions and the rest may align anywhere else in the genome. To restrict the output to exonic sequences, I generated a file containing all the exons plus 10bp at each end for getting splice site information as well. This can be done using the UCSC Table Browser[8]. Choose the hg19 assembly of the human genome and set the track to RefSeq genes and the table to refGene. Use BED format as output format and assign the file an appropriate name (I use target_intervals.bed). By clicking on get output, several more options can be made: Choose ”create one bed record per Exon plus 10bp at each end” and save the file.

Now we have the raw SNP calls and usually a lot of them, but still that's not enough...

8 http://genome.ucsc.edu/cgi-bin/hgTables?command=start

3.2. Variant quality score recalibration

During SNP calling each SNP is assigned a variant quality score which correlates to the possibility of the SNP being called wrong[9]. While the variant score calculated during the SNP calling step is a rough estimate, the variant recalibration step determines a more exact quality score, which can be used to filter out possible artifacts more accurately. However, Variant quality score recalibration needs some known sites to determine different parameters for the Gaussian mixture model used. All of those true sites (coming from HapMap, OmniChip and dbSNP) can be downloaded in the GATK resource bundle[10]. To apply the recalibration two commands are necessary:

```
java -Xmx4g -jar GenomeAnalysisTK.jar \
     -R hg19.fa \
     -T VariantRecalibrator \
     -B:input,VCF snps.vcf \
     -B:hapmap,VCF,known=false,training=true,truth=true,prior=15.0 hapmap_3.3.hg19.sites.vcf \
     -B:omni,VCF,known=false,training=true,truth=false,prior=12.0 1000G_omni2.5.hg19.sites.vcf \
     -B:dbsnp,VCF,known=true,training=false,truth=false,prior=8.0 dbsnp_132.hg19.vcf \
     -an QD -an HaplotypeScore -an MQRankSum -an ReadPosRankSum -an FS -an MQ \
     -recalFile recalfile \
     -tranchesFile output.tranches \
     -rscriptFile output.plots
```

This generates the tranches file which is used for applying the model to the generated SNP calls. The next step looks like this:

```
java -Xmx4g -jar GenomeAnalysisTK.jar \
   -T ApplyRecalibration \
   -R hg19.fa \
   -B:input,VCF snps.vcf \
   --ts_filter_level 99.0 \
   -tranchesFile output.tranches \
   -recalFile recalfile \
   -o snp.vcf.recalibrated
```

The recalibrated SNP calls are then saved to snp.vcf.recalibrated.

3.3. filter SNPs

Although this step is called filtering, I usually don't throw out possible wrong SNP calls and sometimes it proved to be useful to get back to those SNPs in a later step in the analysis. I prefer to flag them according to the reason why they should be filtered. The filtering scheme are partially the recommended ones by the GATK team and some are based on my experience. A SNP which passes through all the filters doesn't necessarily mean a true SNP call and SNPs filtered out don't necessarily define a sequencing artifact, but it gives a clue for possible reasons why a SNP could be wrong.

9 http://www.broadinstitute.org/gsa/wiki/index.php/Variant_quality_score_recalibration
10 http://www.broadinstitute.org/gsa/wiki/index.php/GATK_resource_bundle

```
java -Xmx4g -jar GenomeAnalysisTK.jar \
     -R hg19.fa \
     -T VariantFiltration \
     -B:variant,VCF snp.vcf.recalibrated \
     -o snp.recalibrated.filtered.vcf \
     --clusterWindowSize 10 \
     --filterExpression "MQ0 >= 4 && ((MQ0 / (1.0 * DP)) > 0.1)" \
     --filterName "HARD_TO_VALIDATE" \
     --filterExpression "DP < 5 " \
     --filterName "LowCoverage" \
     --filterExpression "QUAL < 30.0 " \
     --filterName "VeryLowQual" \
     --filterExpression "QUAL > 30.0 && QUAL < 50.0 " \
     --filterName "LowQual" \
     --filterExpression "QD < 1.5 " \
     --filterName "LowQD" \
     --filterExpression "SB > -10.0 " \
     --filterName "StrandBias"
```

I defined some different filters, which show up in the VCF file later-on. The first filter is the SNP cluster filter: So if 3 or more filters are detected within 10 base-pairs (or the number of base pairs you specify in clusterWindowSize) these are flagged as SnpCluster. They are likely to be false positives. The second filter defines SNPs which are HARD_TO_VALIDATE. The filter criterion is stated in the filterExpression argument above: If 4 or more alignments having a mapping quality of MQ0 (which means: it maps to different locations equally well) and the number of alignments which mapped ambiguously are more then a tenth of all alignments, it is hard to decide wether it is an artifact or not.

LowCoverage filter: SNPs which are covered by less than 5 reads may be potential artifacts.

VeryLowQual: SNPs having a SNP quality below 30 are usually artifacts.

LowQual: SNPs having a quality score between 30 and 50 are potential artifacts.

LowQD: the QD flag is defined as „Variant confidence (given as (AB+BB)/AA from the PLs) / unfiltered depth. Low scores are indicative of false positive calls and artifacts.“

StrandBias: SNPs covered only by sequences on the same strand are often artifacts.

All of the flags stated above (MQ0, QD, SB, QUAL, DP) are explained very well in the explanation of the VCF file format of the Unified Genotyper[11].

A very important flag is the AF flag. When dealing with single sample SNP calls, this flag is 0.50 for heterozygous mutations and 1.00 for homozygous mutations. These values can be valuable for the following filter steps.

So now we got a decent SNP calling set with potential artifacts flagged, but still there are some thousands of them. To distinguish important from rather unimportant ones we need some more annotations.

11 http://www.broadinstitute.org/gsa/wiki/index.php/Understanding_the_Unified_Genotyper%27s_VCF_files

4. Annotations using annovar

4.1. Conversion to annovar file format

For annotating SNP calls I use the software annovar[12]. It annotates a lot of different data to the SNPs and is especially suited for exome-level data-sets.

At first we need to convert the VCF file format to the annovar file format. Annovar got it's own script to do that for us[13].

```
convert2annovar.pl --format vcf4 --includeinfo snp.recalibrated.filtered.vcf >
snp.annovar
```

Be sure to include the –includeinfo argument as this will move the annotations from GATK (filters, SNP quality scores and everything else) to the annovar file.

Another script annotates the annovar file. This script needs some annotation files, all of which can be downloaded at their homepage[14]. Be sure to get all the hg19_xxx files if you've done alignment on the hg19 human assembly and save it in the humandb subfolder of the annovar folder.

The script then produces a comma-separated text file with all the annotations, which can be viewed in Excel, OpenOffice Calc or similar programs.

```
summarize_annovar.pl --buildver hg19 snp.annovar ./humandb -outfile snps
```

This creates some files in the current directory. The snps.exome_summary.csv contains all annotated exonic SNPs and should contain the filter information we obtained in the prior steps.

The columns in the output file should include:

- Function (exonic or splicing)
- Gene
- ExonFunction (whether it is a synonymous SNV, stopgain, non-/frameshift substitutions[15]
- Amino-acid change
- Conservation
- Segmental Duplications (the closer to 1 the more similar other parts within the genome are)
- 1000genome allele frequencies (the default for hg19 is the november 2010 release), this one is released in three equal columns
- dbSNP reference number
- prediction scores: annovar outputs precomputed scores of different prediction programs (SIFT, Polyphen2, LRT, MutationTaster and a conservation Score (PhyloP))[16].
- Position and the additional GATK information

12 http://www.openbioinformatics.org/annovar/
13 http://www.openbioinformatics.org/annovar/annovar_download.html
14 http://www.openbioinformatics.org/annovar/annovar_download.html
15 Annovar calls Indels substitutions, which is somehow misleading...
16 Additional information in Liu et al. „dbNSFP: A lightweight Database of Human Nonsynonymous SNPs and Their Funtional Predictions.“ 2011 in Human Mutation

So that's it..., nearly. Now comes the really tough part: the search for the right mutation in the vast number of different variations which may or may not be benign. Through the Excel Data filtering you could get rid of different SNPs which you think might be wrong ones: dbSNP SNPs, SNPs frequent in 1000 genome SNPs, SNPs within segmental duplications, synonymous SNPs, possible artifacts, heterozygous or homozygous mutations or any other annotated information. There is no general rule how to do that, as that varies from experiment to experiment, the number of samples or the expected outcome.

As the comma-separated text file can be opened with Excel this information can easily be handed to the experienced biologist for letting them filter themselves...

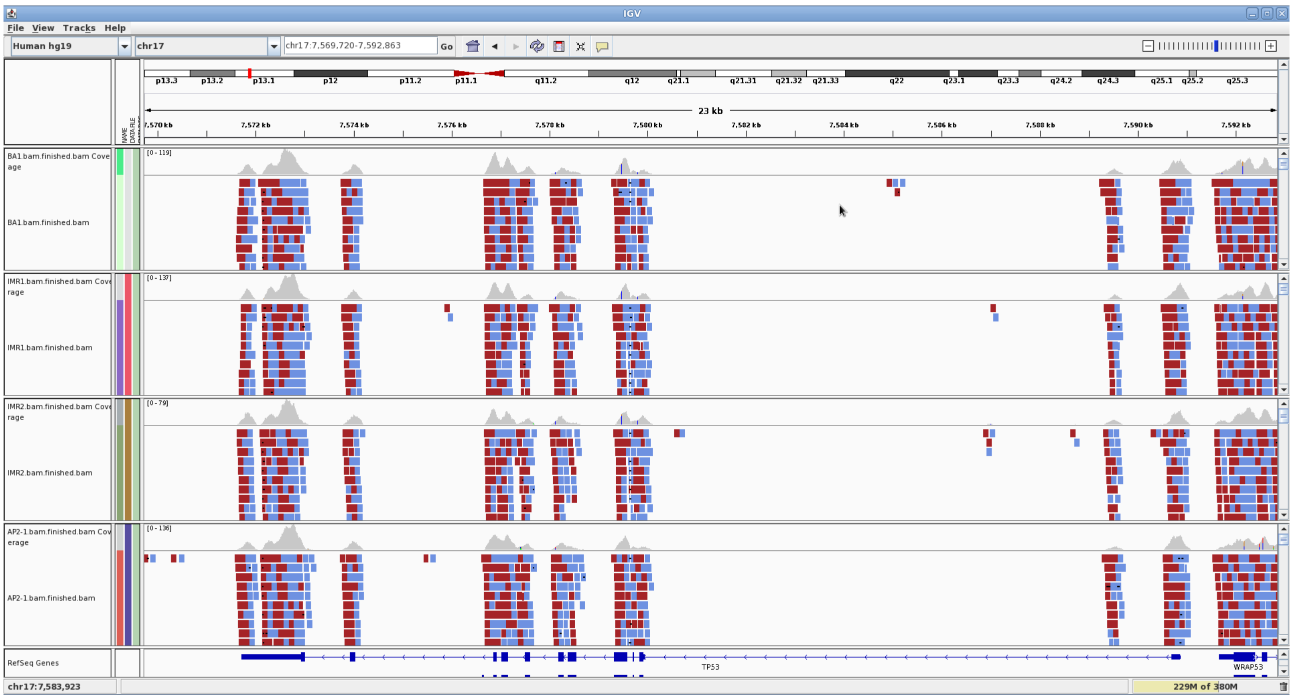
5. Visualization

Most biologists are not too keen to handle large text-based lists of SNPs, stuffed with various values they never heard of. A nice way to present NGS-data is visualization. Moreover, visual analysis of SNPs often helps to judge between sequencing artifacts and true variants and comparison between different datasets of different samples at the same site is very easy.

A good tool for NGS visualization is IGV[17]. The code is in Java and it supports a lot of data formats which can be displayed along the exome data.

So download IGV[18] and start it by double clicking on the starting script (igv.sh in my case, igv.bat for Windows users). Select the right reference assembly and load some alignment (.bam) files. Be sure to take the realigned and recalibrated ones to have it displayed correctly.

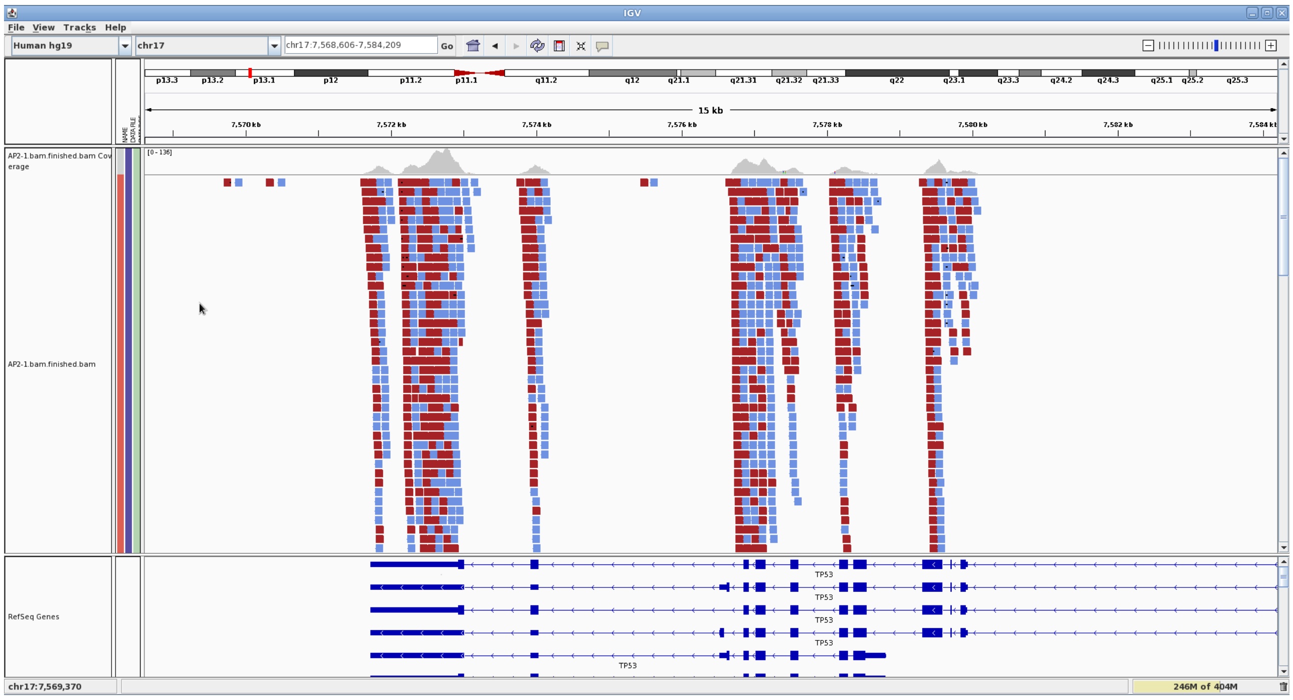
The refGene annotation is already loaded, but several other annotations can be loaded as well (e.g. dbSNP data).



Visualization using IGV. Four different samples are loaded and refGene annotations are displayed on the bottom.

17 http://www.broadinstitute.org/software/igv/
18 http://www.broadinstitute.org/software/igv/download

Alignments in IGV are displayed as colored bars (see figure below) spanning a certain chromosomal range. Above the alignment section there is the coverage section, which shows a histogram of the read depth of the position. For exome data most reads should align at exonic sequences through targetted enrichment of those regions (which can be seen in the figure below).



Visualization of datasets helps to analyse:

- Quality of the dataset (average coverage, enrichment specificity, number of random errors...)
- Quickly look at specific genes or regions
- Compare multiple datasets
- Present data to colleagues

6. Glossary

BAM	Binary Alignment / Map format
BED	Browser extensible Data format
BWA	Burrows-Wheeler Alignment
FastA	FastAlignment format
GATK	Genome Analysis Toolkit
Indel	Insertion and / or deletion
PCR	Polymerase chain reaction
SAM	Standard Alignment/ Map format
SNP	Single (or Simple) nucleotide polymorphism
VCF	Variant call format