

Variant Calling from Genomic Sequencing Data

EMBL Advanced Course 2011
EMBL Heidelberg

Course Materials

Tobias Rausch
July 2011

Contents

1	Introduction	3
1.1	Getting Started	3
1.2	Data Description	3
2	Read Depth	4
2.1	Alignment Statistics	4
2.2	Detecting Large-Scale Aberrations	5
3	SNPs and Short Indel Calling	10
3.1	SNPs	10
3.2	SNP Filtering and Annotation	10
3.3	SNP Annotation using Annovar	13
4	Structural Variant Calling	14
4.1	Detecting Structural Variants	14
4.2	Large Deletions	14
4.3	Tandem Duplications	16
5	Genome Capture	19
5.1	Baits and Target Regions	19
5.2	Size Distribution	19
5.3	On-target / Off-target	20
5.4	GC-Content	23
6	Assembly	26
6.1	De novo Assembly	26
6.2	Reference-based Assembly	26

Introduction

1.1 Getting Started

The log-in to the machine in front of you is:

```
username: teachZ
password: Te@achZ
```

Please replace the Z with your unique group number that we distribute in the course. To speed up some of the computations during the course we first make a local working directory, where we can copy the raw data.

```
cd /tmp/
mkdir guestZ
cd guestZ
```

1.2 Data Description

During the whole course we will work with 4 samples coming from the same individual. One sample is a germline sample that we are going to use as a control. The remaining 3 samples are tumor samples. The raw data is available in `/g/solexa/RunVol10/Runs/scratch`, where `g1` is the germline sample and `t1`, `t2` and `t3` are the 3 tumor samples. All 4 samples have been sequenced to 30x using whole-genome short-read sequencing technology. To save time, all the sequenced data has been mapped already to the latest human genome reference sequence, termed `hg19` or synonymously `GRCh37`. The reference sequence is available as a fasta file in `./scratch/ref`. The alignments are provided in the `./scratch/data` folder in BAM format. Since this is a binary format please make sure you have read the brief SAM/BAM tutorial before starting the exercises below. All 4 data sets are confidential. You are not allowed to use the data outside of this course.

Read Depth

2.1 Alignment Statistics

Calling somatic genomic alterations on a full 30x genome would take several hours if not days. Because of that, we have limited the data analysis part of this course to chromosome 17 and chromosome 8. The size of chr8 is 146,364,022 bp, the size of chr17 is 81,195,210 bp. Before starting the actual variant calling, let us first compute some basic alignment statistics using the samtools flagstat command (Li et al., 2009) and some simple shell commands.

1. How many reads are mapped to chr17 for each sample? How many reads are properly paired for each sample?

```
samtools flagstat ../data/g1.chr17.bam
samtools flagstat ../data/t1.chr17.bam
samtools flagstat ../data/t2.chr17.bam
samtools flagstat ../data/t3.chr17.bam
```

2. Extract the insert size of all properly mapped pairs (column 9 in SAM format) for one of the samples. Make a histogram of these insert sizes using R. An example of such a plot is shown in Figure 2.1 and Figure 2.2.

```
samtools view ../data/g1.chr17.bam |
awk 'and($2, 0x0002) && and($2, 0x0040)' |
cut -f 9 | sed 's/^-//' > inserts.txt
```

3. Create some simple statistics about the insert size distribution using the R commands mean, median, range, sd and boxplot.

```
x = scan("inserts.txt")
hist(x)
mean(x)
median(x)
range(x)
sd(x)
boxplot(x)
```

The number of mapped reads for each sample can be used to normalize for differential read counts when plotting read depth ratios. The insert size distribution is crucial for calling structural variants and we will revisit that issue later on in Chapter 4.

2.2 Detecting Large-Scale Aberrations

A very simple method to highlight large-scale chromosomal aberrations uses a read-depth approach. We first compute the coverage for each sample in a window-by-window fashion. The non-overlapping windows are of size 10kb. This cannot be done anymore using shell commands, so I have written a tool (Rausch, 2010) that calculates the coverage for a given set of genomic intervals or genome-wide in a window-by-window manner called `cov`.

```
cov -g ../ref/chr17.fa ../data/g1.chr17.bam > g1.cov
cov -g ../ref/chr17.fa ../data/t1.chr17.bam > t1.cov
cov -g ../ref/chr17.fa ../data/t2.chr17.bam > t2.cov
cov -g ../ref/chr17.fa ../data/t3.chr17.bam > t3.cov
head g1.cov
```

Once the program is finished each 10kb window of chr17 has been annotated with the number of reads falling into that window. If we sum up the last column we should get the total number of reads for each sample minus the filtered redundant reads.

```
awk '{SUM+=$4} END {print SUM;}' g1.cov
awk '{SUM+=$4} END {print SUM;}' t1.cov
awk '{SUM+=$4} END {print SUM;}' t2.cov
awk '{SUM+=$4} END {print SUM;}' t3.cov
```

You can compare this output with the previously computed sequencing statistics using `samtools flagstat`. We are now using R to plot the \log_2 ratio for each non-overlapping window comparing one of the tumor samples with one of the normal samples.

```
R
x = read.table("g1.cov")
y = read.table("t1.cov")
plot(log2(y[,4] / x[,4]))
plot(x[,2], log2(y[,4] / x[,4]))
```

So in total we are plotting t1 against g1, t2 against g1 and t3 against g1. The plots are shown in Figure 2.3, Figure 2.4 and Figure 2.5.

1. What kind of chromosomal aberrations can you see?

01006_SN169_0174_B80223ABXX/Data/Intensities/BaseCalls/GERALD_16-1
Pairs in plot: 19535450 out of 19545799 (0.9995)
Overall median: 249

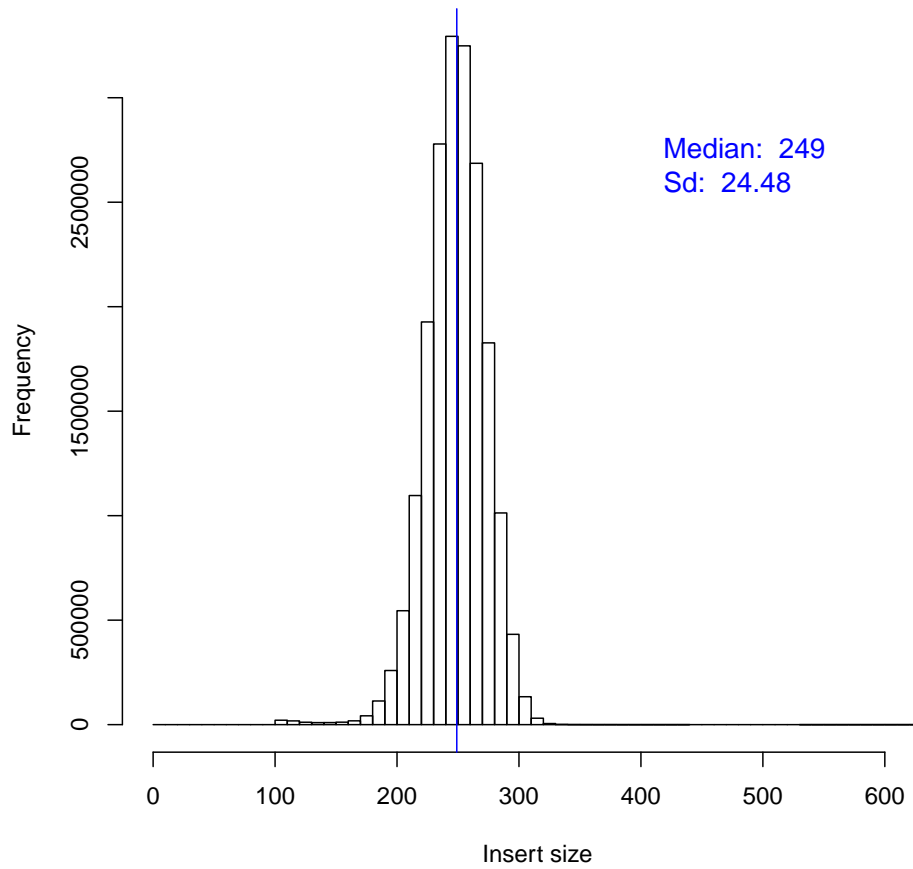


Figure 2.1: Insert size distribution for a paired-end library.

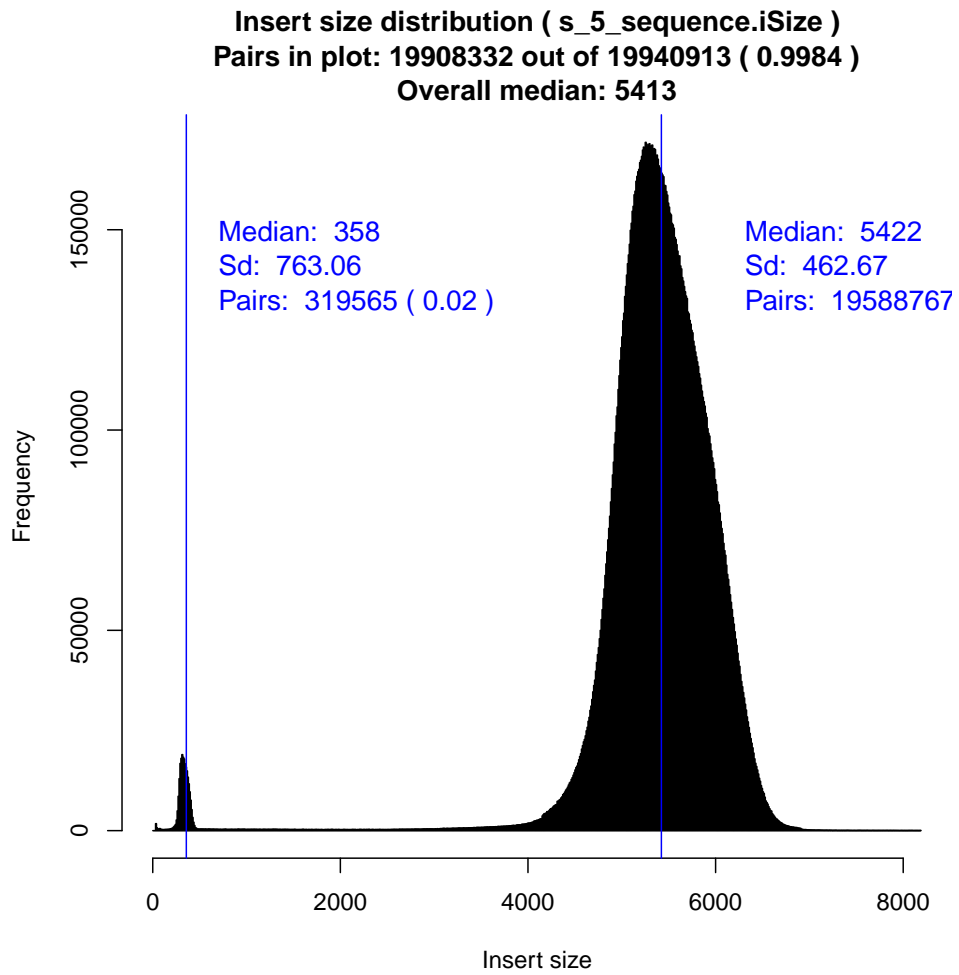


Figure 2.2: Insert size distribution for a mate-pair library.

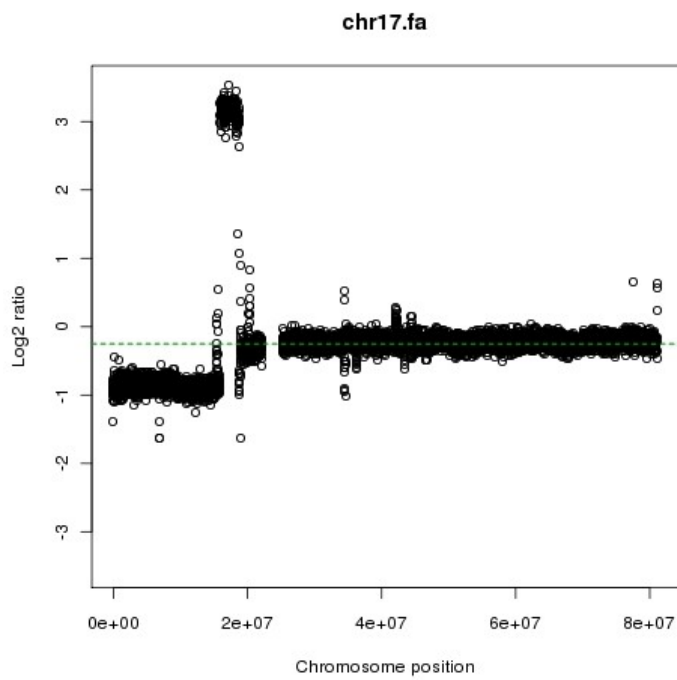


Figure 2.3: Read-depth plot of t1 against g1.

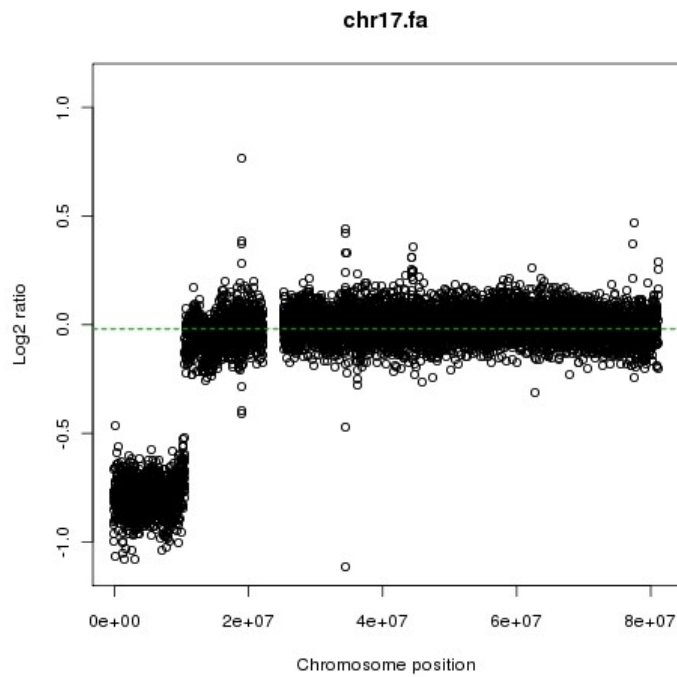


Figure 2.4: Read-depth plot of t2 against g1.

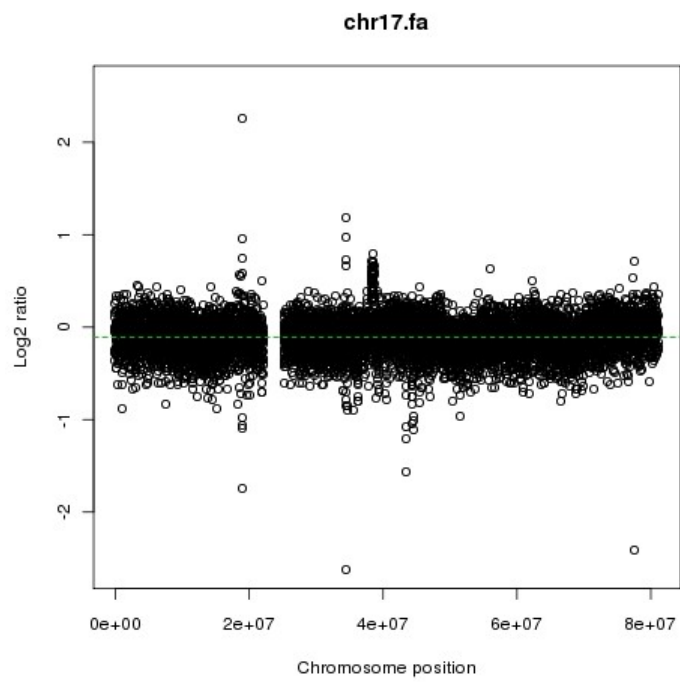


Figure 2.5: Read-depth plot of t3 against g1.

SNPs and Short Indel Calling

3.1 SNPs

Calling SNPs on a full 30x human genome sample takes several hours. Because of that, we restrict the calling in this course to chr17. For SNP calling, the samtools package offers the so-called mpileup command. Please use mpileup on all 4 samples, g1, t1, t2 and t3. Each command should take about 45 minutes for alignments restricted to chr17.

```
samtools mpileup -ugf ../ref/chr17.fa ../data/g1.chr17.bam |  
  bcftools view -bcvg - > g1.bcf  
bcftools view g1.bcf |  
  perl vcfutils.pl varFilter -D 100000000 > g1.vcf
```

The columns of that file are briefly explained in Table 3.1. The samtools package also has a simple alignment viewer called samtools tvview. In that viewer, a dot or a comma in the read stands for a match to the reference, either on the forward (dot) or reverse (comma) strand. Any other DNA nucleotide that appears in the read stands for a mismatch. Upper case letters are mismatches on the forward strand whereas lower case letters are mismatches on the reverse strand.

1. Pick one of the called SNPs and compare the mpileup output with the graphical view of the alignment using samtools tvview. A screenshot of tvview for one of the SNPs is shown in Figure 3.1.

```
head -n 12020 g1.chr17.snps | tail -n 40  
samtools tvview ../data/g1.chr17.bam ../ref/chr17.fa
```

2. Do the same thing for one of the called indels.
3. How many variants have been called in total for each sample?
4. Please have a look at one of the coding SNPs on chr17 at position 7577539. Is that a silent mutation? If not, what kind of amino acid change does this SNP introduce in the TP53 gene?

3.2 SNP Filtering and Annotation

Obviously, a significant amount of variant sites are false positives. For instance, multiple SNPs occurring right next to each other are almost always due to misalignments. Likewise

Index	Field Name	Description
1	CHROM	Chromosome identifier
2	POS	1-based position. For an indel, this is the position preceding the indel.
3	ID	Variant identifier. Usually the dbSNP rsID.
4	REF	Reference sequence at POS involved in the variant. For a SNP, it is a single base.
5	ALT	Comma delimited list of alternative sequence(s).
6	QUAL	Variant quality.
7	FILTER	Semicolon delimited list of filters that the variant fails to pass.
8	INFO	Semicolon delimited list of variant information.
9	FORMAT	Colon delimited list of the format of individual genotypes in the following fields.
10+	Sample(s)	Individual genotype information defined by FORMAT.

Table 3.1: Brief summary of the Mpileup vcf format.



Figure 3.1: SNP displayed in samtools tview.

huge pileups of reads usually signal a collapsed repeat where inter-repeat differences show up as a putative SNPs. Given the large amount of called SNPs, we also lack a proper SNP annotation that allows us to quickly identify coding missense or nonsense mutations. Doing such a manual check-up like you just did for the TP53 mutation is a cumbersome task. Because of all these limitations many sophisticated SNP callers have been developed in the past. In our lab, we are currently using Annovar (Wang et al., 2010) and the Genome Analysis Toolkit (McKenna et al., 2010) (GATK) to annotate and filter SNPs. Since the annotation using GATK is very time-consuming, I have precomputed this SNP

3. SNPs and Short Indel Calling

annotation before the course and placed the SNP calls of the Genome Analysis Toolkit of chr17 into the data folder.

1. How many SNPs have been called by the GATK?

```
cat ../data/t1.chr17.snps | cut -f 1-5 | sort | uniq | wc -l
```

2. How many SNPs are novel, how many are present in dbSNP?

```
cat ../data/t1.chr17.snps | awk '$3=="."' | cut -f 1-5 |  
sort | uniq | wc -l
```

3. How many SNPs are intronic, how many are coding?

```
cat ../data/t1.chr17.snps | grep "intron" | cut -f 1-5 |  
sort | uniq | wc -l
```

4. How many SNPs are missense or nonsense mutations?

```
cat ../data/t1.chr17.snps | grep "CDS" | cut -f 1-5 |  
sort | uniq | wc -l
```

Let's focus for the time being on the novel missense and nonsense mutations.

```
awk '$3=="."' ../data/g1.chr17.snps | egrep "missense|nonsense" > g1s  
awk '$3=="."' ../data/t1.chr17.snps | egrep "missense|nonsense" > t1s  
awk '$3=="."' ../data/t2.chr17.snps | egrep "missense|nonsense" > t2s  
awk '$3=="."' ../data/t3.chr17.snps | egrep "missense|nonsense" > t3s
```

Given these novel missense or nonsense mutations let's run some simple cross comparisons.

1. How many SNPs are left per sample? You should take care of SNPs occurring in multiple transcripts.

```
cut -f 1-5 g1s | sort | uniq > g1.snp.txt  
cut -f 1-5 t1s | sort | uniq > t1.snp.txt  
cut -f 1-5 t2s | sort | uniq > t2.snp.txt  
cut -f 1-5 t3s | sort | uniq > t3.snp.txt  
wc -l *.snp.txt
```

2. How many are shared among all samples?
3. How many of these are unique to a sample?
4. Have a look again at the shared TP53 mutation. In which samples has it been called homozygous or heterozygous? Any ideas how this can happen if all samples belong to the same individual?
5. How would you validate a SNP and is that necessary? How do you expect a heterozygous SNP to look like in a chromatogram? We provide one example chromatogram, try to find the heterozygous SNP in the tumor and check that position in the normal tissue?
6. Is there a way to rank the individual SNP calls based upon their putative mutational consequences?

3.3 SNP Annotation using Annovar

Annovar cannot yet use SNP calls in vcf format. Therefore we first convert the vcf file to Annovar's input format.

```
convert2annovar.pl -format vcf4 g1.vcf > g1.annovar
sed -i 's/\.fa\t\t/' g1.annovar
```

For annotation purposes, Annovar relies on the annotation database of the UCSC browser or alternatively user's can provide their own annotation files in bed or vcf format. We are obviously most interested in classification of coding and non-coding variants so we first download the refGene database.

```
mkdir hg19db
annotate_variation.pl --buildver hg19 --downdb seq hg19db/hg19_seq
retrieve_seq_from_fasta.pl hg19db/hg19_refGene.txt -seqdir
hg19db/hg19_seq -format refGene -outfile hg19db/hg19_refGeneMrna.fa
```

This builds the mRNA sequences from the whole genome sequence files. For hg19, UCSC already supplies mRNA sequences so we can directly download these mRNA sequences.

```
annotate_variation.pl --buildver hg19 --downdb refGene hg19db/
```

Using these mRNA sequences, we can quickly annotate synonymous and nonsynonymous SNP sites.

```
annotate_variation.pl --buildver hg19 --outfile g1
--hgvs g1.annovar hg19db/
```

This simple annotation pipeline can be used for any species in UCSC by simply replacing hg19 with, for instance, dm3 (*D. melanogaster*) or mm9 (Mouse).

1. How many variants are intergenic, intronic and exonic?
2. How many synonymous and nonsynonymous variants have been called for each sample?
3. What is the overlap between the GATK SNP Calls and the Mpileup SNP Calls?

For human data, Annovar can also annotate segmental duplications, transcription factor binding sites, dbSNPs and SIFT and Polyphen scores. To annotate, for instance, the SNPs present in dbSNP we can use the following.

```
annotate_variation.pl -buildver hg19 -downdb snp132 hg19db/
annotate_variation.pl --buildver hg19 --filter --dbtype snp132
g1.annovar hg19db/
```

Structural Variant Calling

4.1 Detecting Structural Variants

The most prominent method to detect large structural variants is paired-end mapping (Korbel et al., 2009). The basic idea of this method is to extract all reads having an abnormal mapping distance or an abnormal orientation. Subsequently, these reads are clustered based on regional coherence. For structural variants introducing a copy number change such as deletions or tandem duplications, one also seeks read-depth support. In addition, the next-generation sequencing technologies are constantly improving on the number of bases per read. The longer the reads the more fruitful is also the detection of split-reads to further support a paired-end structural variant call. This has the additional benefit of mapping structural variants at breakpoint resolution. Recently, our lab focused on combining paired-end, read-depth and split-read alignment methods to increase sensitivity and specificity of structural variant calls.

1. How can the insert size distribution be used to identify cutoffs for the classification of paired-ends?
2. What is the difference between sequencing and spanning coverage? Why is the later more important for paired-end structural variant discovery?

4.2 Large Deletions

We recently developed a paired-end structural variant detection tool called delly. For each raw paired-end call we then try to identify split-reads.

1. Please run delly on chr8 for all samples.

```
delly -pi g1 -g ../ref/chr8.fa -o g1.pe -b g1.br ../data/g1/*.bam
delly -pi t1 -g ../ref/chr8.fa -o t1.pe -b t1.br ../data/t1/*.bam
delly -pi t2 -g ../ref/chr8.fa -o t2.pe -b t2.br ../data/t2/*.bam
delly -pi t3 -g ../ref/chr8.fa -o t3.pe -b t3.br ../data/t3/*.bam
emacs -nw g1.br
emacs -nw g1.pe
```

The raw paired-end deletion calls are written to g1.pe, t1.pe, t2.pe and t3.pe, respectively. For a subset of these calls split-reads could be found. These split-read supported paired-end calls are in the files g1.br, t1.br, t2.br and t3.br, respectively.

2. Please have a look at the paired-end and split-read call files. Each call starts with a print out of all supporting pairs or split-reads and finally, there comes a summary line. For the paired-end calls, the summary line contains the chromosome as well as the estimated start and end of the deletion. This information is succeeded by the size of the deletion, the number of supporting pairs and a unique deletion id. For the split-reads, the summary line contains the chromosome as well as the exact breakpoint coordinates of the deletion and the exact size. This information is succeeded by the number of split-reads, the alignment quality of the flanking sequence and the unique deletion id. That deletion id is followed by the sequence of the left and right breakpoint as well as the length of the microinsertion and microhomology if present.

As you may have noticed we pass each bam file separately to delly. For the current data set, this is not so important since all bam files are coming from the same library. If you, however, start to combine mate-pair and paired-end libraries this is crucial since each of the libraries has a different insert size.

1. How many deletions have been identified by paired-end mapping?

```
cat t1.pe | grep ">Deletion" | wc -l
```

2. How many calls could be substantiated with split-reads?

```
cat t1.br | grep ">Deletion" | wc -l
```

3. What are potential problems that could lead to a false positive paired-end call?
4. What are potential problems that could lead to a false positive split-read call?
5. Take one of the split-read supported calls and blast the reads individually? Do you find alternative mappings that would invalidate the split-read alignment?
6. Please use the UCSC genome browser to investigate some of the split-read supported calls. Can you identify any type of mutational events (NHR, NAHR, retrotransposition, Alu, Line) that may have formed these structural variants?

To answer the last question it might be helpful to switch on the following UCSC browser tracks: RefSeq Genes, DGV Struct Var, Repeat-Masker, Simple Repeats and Microsatellites. At least one of the events is several kb in size and encompassing some exons.

1. Can you spot that large deletion already in one of the read-depth plots of chromosome 8 shown in Figure 4.1, Figure 4.2 and Figure 4.3?
2. Given that a deletion affects a gene, what potential functional effect may this have?

Once a somatic alteration of interest has been found, we usually seek further validation of such a structural variant event.

1. How can a deletion be validated using PCR? What is important regarding the primer design?
2. We validated the large deletion using Sanger sequencing. Can you spot the breakpoint in the chromatogram using the traceedit viewer?

```
./traceedit.sh
```

4.3 Tandem Duplications

Using paired-end mapping, we can also detect tandem duplications.

1. What kind of mapping pattern characterizes tandem duplications?
2. Do we expect a read-depth signal?

For tandem duplications one could also try to detect split-reads but at the moment we have not yet done so. We do, however, have implemented a simple paired-end caller that you can run on all samples.

```
duppy -i g1 -o g1.dup ../data/g1/*.bam
duppy -i t1 -o t1.dup ../data/t1/*.bam
duppy -i t2 -o t2.dup ../data/t2/*.bam
duppy -i t3 -o t3.dup ../data/t3/*.bam
```

1. How many tandem duplications have been identified by paired-end mapping?
2. Is there any large duplication supported by more than 10 pairs of high mapping quality (>100) that is viewable in the log₂ ratio plots of Figure 4.1, Figure 4.2 and Figure 4.3?
3. We haven't discussed so far inversions and translocations. Can these be identified with paired-ends? Do you expect a read-depth signal?

There is one final last question regarding the calling of large structural variants.

- What are the individual strength and weakness of paired-end mapping, split-read alignment and read-depth to detect structural variants?

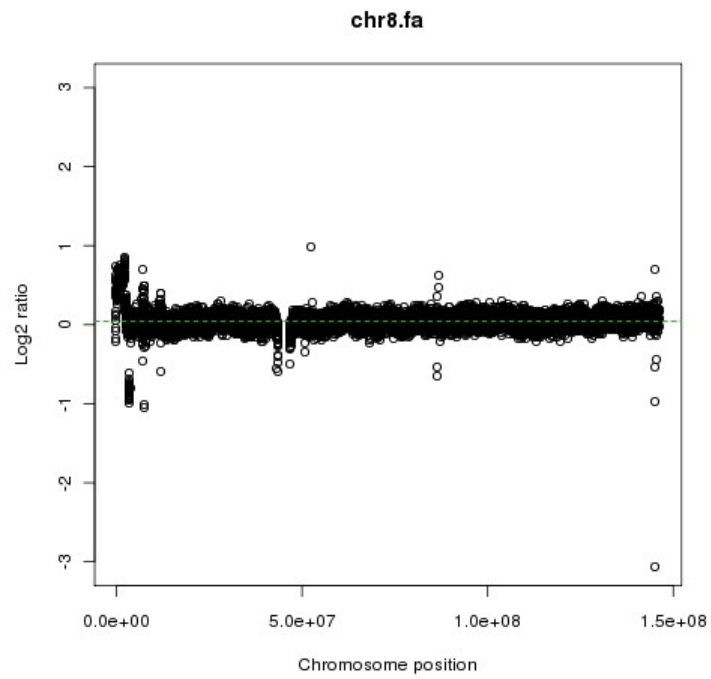


Figure 4.1: Read-depth plot of t1 against g1.

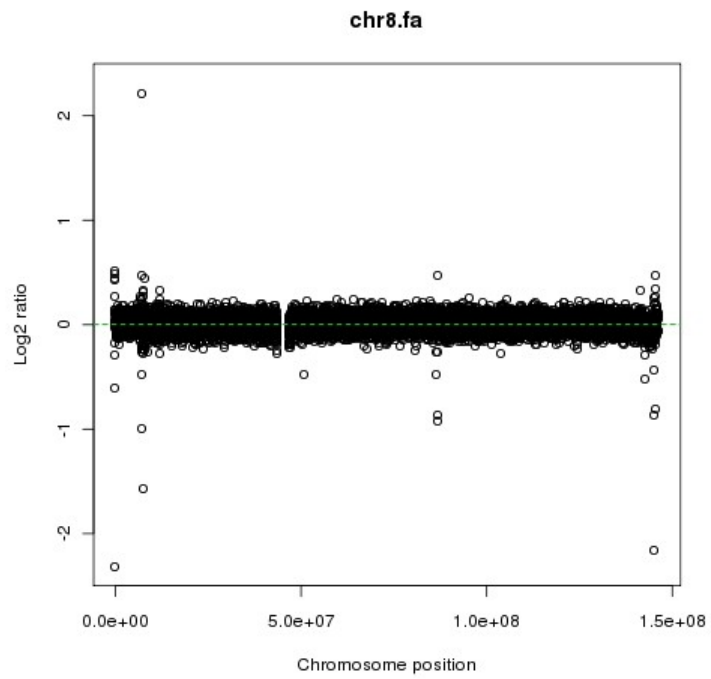


Figure 4.2: Read-depth plot of t2 against g1.

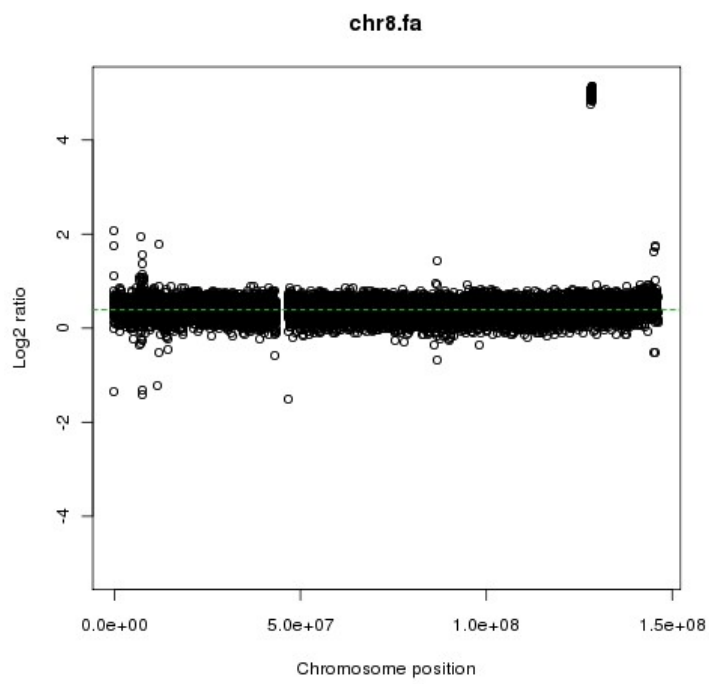


Figure 4.3: Read-depth plot of t3 against g1.

Genome Capture

5.1 Baits and Target Regions

Agilent kindly provided two bed files, one that contains all the bait coordinates and one that contains all the target region coordinates. Let us have a look at these two files first.

```
cd /tmp
mkdir targets
cp /g/solexa/RunVol10/scratch/targets/*.bed /tmp/targets/
cd /tmp/targets
ls
head baits.bed
head targets.bed
```

We now use the UCSC genome browser (genome.ucsc.edu) to visualize the bait and target regions for an arbitrary gene. I picked the ADCK4 gene on chr19 and extracted all baits and targets close to it. Using simple Linux command you can extract the regions for any gene you like. For instance, for the ADCK4 gene, I used:

```
awk '$1=="chr19" && $2>=45882982 && $3<=45920732' baits.bed
awk '$1=="chr19" && $2>=45882982 && $3<=45920732' targets.bed
```

When you open up the UCSC genome browser pick the hg18 version and press submit. Then click manage custom tracks, add the two provided bed files for the ADCK4 gene and go back to the genome browser. Zoom in for one of the exons to get an impression how a single exon is covered by baits and target regions.

5.2 Size Distribution

Let us try to get a rough impression how much of the genome we tried to capture. To do this we simply have to sum up the lengths of all target regions. With awk that is a piece of cake.

```
wc -l targets.bed
awk '{SUM+=$3-$2+1} END {print SUM;}' targets.bed
```

1. What is the total number of bases included in all target regions?
2. What is the size of that region compared to the total length of the human genome?

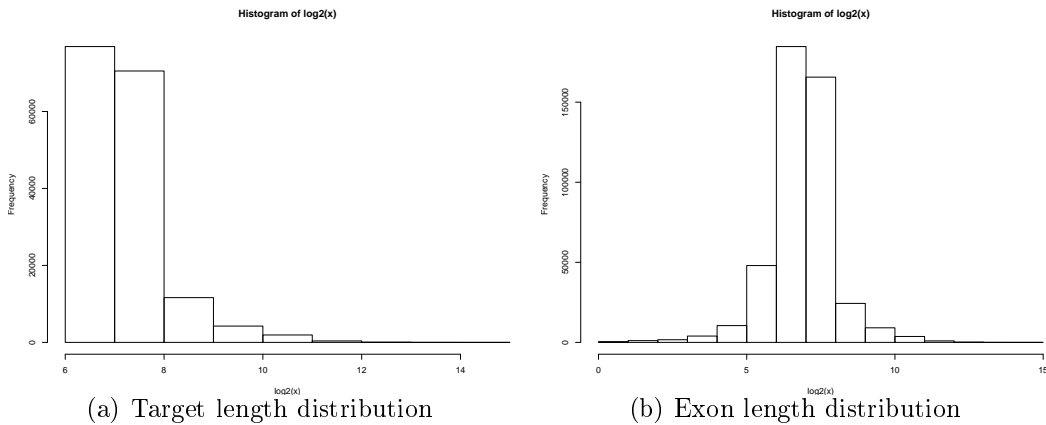


Figure 5.1: Comparison of length distribution of exons and targets

We can also plot the lengths of all target regions with R to get an impression of the distribution of these target lengths (see Figure 5.1(a)).

```
awk '{print $3-$2+1;}' targets.bed > targetlen.txt
R
```

Now we create a simple histogram of the lengths with R.

```
x = scan("targetlen.txt")
hist(log2(x), breaks=10)
```

Obviously, each target region is at least as long as a single bait. This is not so telling by itself but we can do the same for a list of all exons in the human genome (see Figure 5.1(b)).

1. Make a histogram of the lengths of all exons in the file exon.bed.
2. Compare the exon histogram with the target region histogram. What should you be aware of when you analyze the on-target reads?

If you sum up the lengths of all exons you notice that this sum is much larger than the total length of all target regions. This is due to alternative splicing. Hence, the same exon or overlapping exons appear multiple times in the file. In order to determine the on-exon statistic we need a file with the unique exonic coordinates: basically a map for each base stating if that base is part of an exon or not. This is a bit more tricky and cannot be done easily with simple shell commands (So finally there is a need for a mediocre computer scientist like me). The file exon_uniq.bed has the non-overlapping exonic regions. If you sum up all regions present in this file you should get a slightly smaller value than the one for the target regions. The reason for the difference are targeted miRNAs and exons shorter than the target region containing it.

```
awk '{SUM+=$3-$2+1} END {print SUM;}' exon_uniq.bed
```

5.3 On-target / Off-target

It is time to get hold of some real data. Please copy the files to your local tmp directory. This may take a while because the files are very large. However, the subsequent processing will be quicker since the files are on your local disk then.

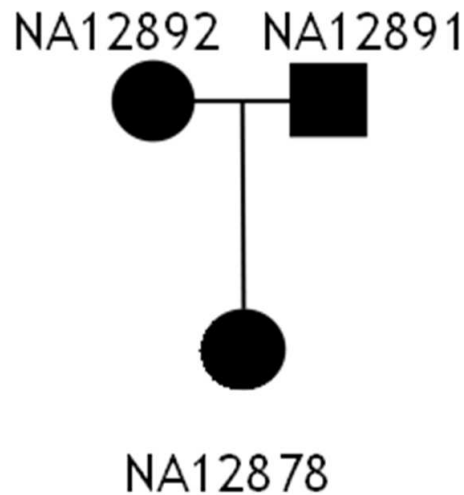


Figure 5.2: HapMap Trio.

```
cd /tmp
mkdir illumina
cp /g/solexa/RunVol10/scratch/target/illumina/*.bam /tmp/illumina/
cd /tmp/illumina
ls
```

There are two BAM files, one for the father and one for the daughter from the HapMap trio (see Figure 5.2). We can easily get all reads in a certain window using awk.

```
samtools view NA12878.bam | awk ' $3=="chr1" && $4>=58932 && $4
<=59892'
```

For all windows this is a bit more time-consuming and I have prepared a coverage file showing the read count for each target region and each exonic region.

```
cp /g/solexa/RunVol10/scratch/target/illumina/*.cov /tmp/illumina/
head NA12878.targets.cov
```

Each region has been annotated by two additional values in the last two columns. The first value is the number of mapped bases and the second value is the number of reads falling in that region. We can quickly sum up the last column to get an idea of how many reads are on target.

```
awk '{SUM+=$6} END {print SUM;}' NA12878.targets.cov
awk '{SUM+=$6} END {print SUM;}' NA12891.targets.cov
```

We need to put that number into relation to the total number of reads. Luckily, the samtools provide this kind of sequencing statistics using the flagstat command.

```
samtools flagstat NA12878.bam
samtools flagstat NA12891.bam
```

You should see a row stating the number of mapped reads. Use that number and the total number of reads in the target regions to estimate the percentage of reads on-target. You can easily render this information as a pie chart using R (see Figure 5.3 for NA12878).

```
mapped=38163843
```

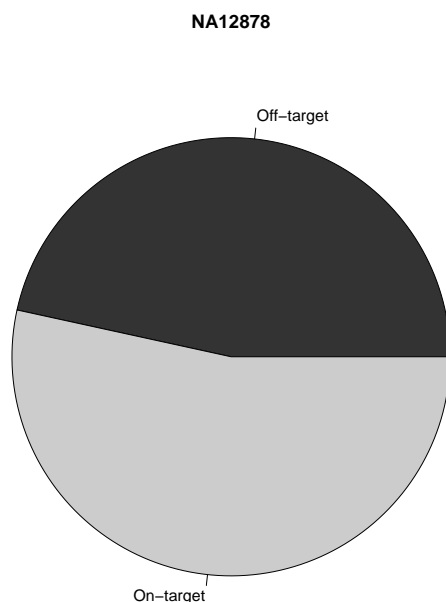


Figure 5.3: Pie chart of on-target reads for NA12878.

```
target=20389403
pie(c(mapped-target , target), c(" Off-target ", "On-target "), main="
  NA12878", col=gray(c(0.2 , 0.8)))
```

You can do the same now for the on-exon and off-exon analysis.

1. Compute the number of reads in exonic regions using both *.exon.cov files.
2. What is the percentage of reads in exonic regions compared to all mapped reads?

I did this on-target / off-target and on-exon / off-exon analysis for all Illumina and SOLiD data sets. The results are summarized in Figure 5.4. If you want to create that plot yourself just create a matrix in R with 3 rows for mapped, on-target and on-exon read counts. Out of this matrix you can easily create the dot chart shown in Figure 5.4.

```
x = c(133915955, 79815980, 66922127, 108092260, 62078729, 51909349,
      38163843, 20389403, 16196544, 29133027, 16259993, 12939470)
mat = matrix(x, nrow = 3, byrow = FALSE)
colnames(mat) = c("NA12878-SOLiD", "NA12891-SOLiD", "NA12878-Illumina",
                  "NA12891-Illumina")
rownames(mat) = c("Mapped", "On-target", "On-exon")
dotchart(mat/1000000, xlab="#Million Reads")
```

Well, there are, of course, some more advanced research questions.

1. How many targets have not a single mapped base?
2. How many of these unmapped targets are equal among all individuals? What might be the reason that these targets are unmapped in all individuals?
3. How uniform have the targets been captured? Calculate the average coverage for each target and make a histogram out of all these values.

Feel free to try your own approach to answer these questions. Here is what I did. The targets without any mapped bases can be easily counted using `wc` and `awk`.

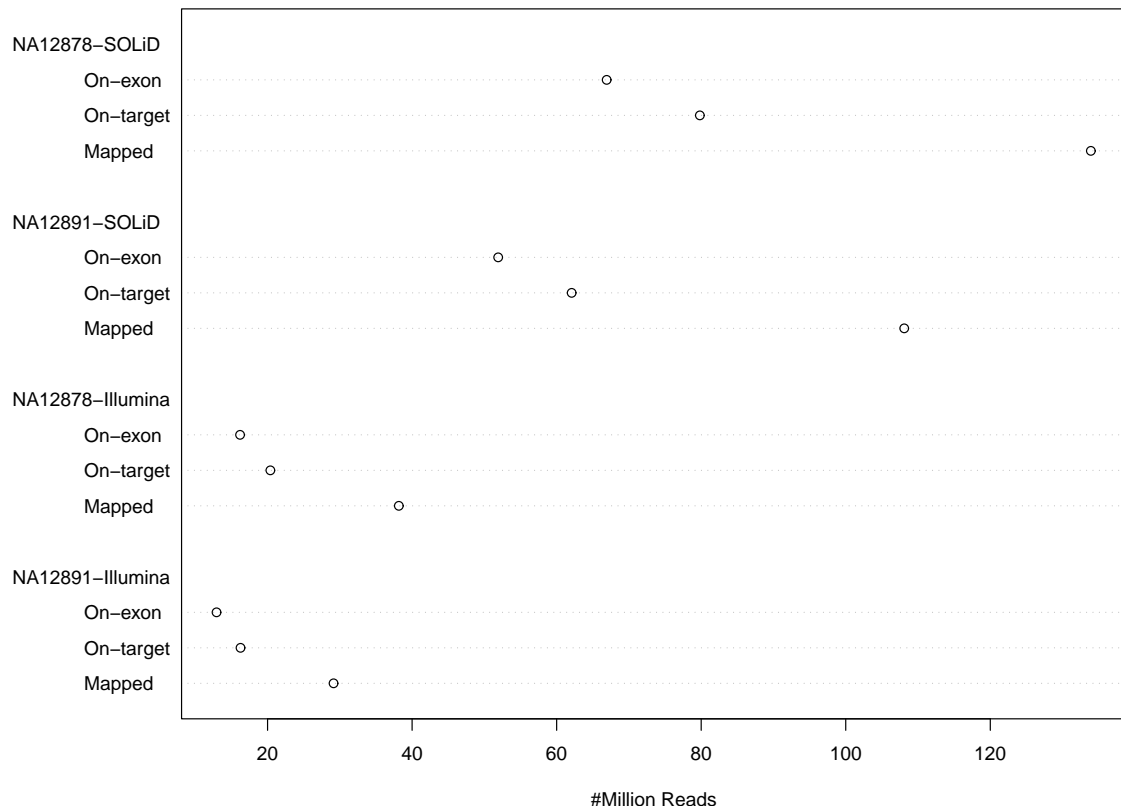


Figure 5.4: Dot chart of all mapped, on-target and on-exon reads.

```
awk '$5==0' NA12878.targets.cov > NA12878.unmapped
awk '$5==0' NA12891.targets.cov > NA12891.unmapped
cat NA12891.unmapped | wc -l
```

To compare how many targets have not been mapped in NA12878 and NA12891 we can use sort and uniq.

```
sort *.unmapped | uniq -d | wc -l
```

The histogram of the coverage values is a bit more involved. We first calculate the average coverage using the number of mapped bases divided by the target region size.

```
awk '{print $5/($3-$2+1);}' NA12878.targets.cov > avg.cov
```

Then we start R and plot the histogram (see Figure 5.5).

```
x=scan("avg.cov")
hist(log2(x))
```

5.4 GC-Content

As you have noticed the coverage is not uniform across all targets. One contributing factor to this non-uniform coverage is the gc-content. In Figure 5.6 I have plotted the gc-content distribution of all target regions. I have created a file targets.gc containing the gc-content of each target region. We are now going to plot the gc-content of a target against its average coverage. Please copy the following files.

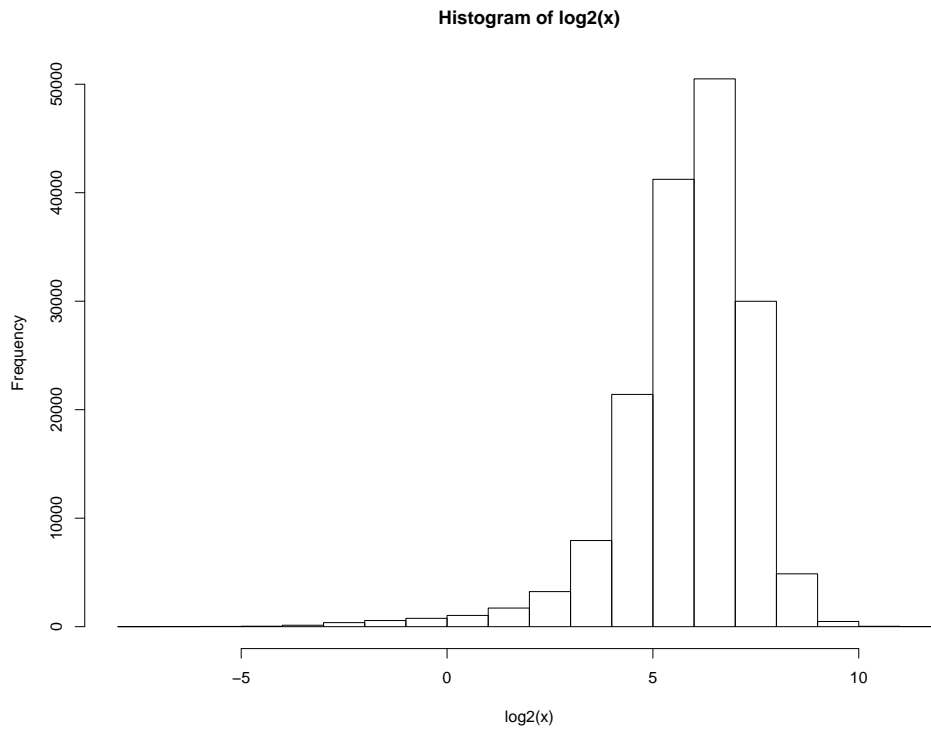


Figure 5.5: Histogram of the average coverage of each target.

```
cd /tmp
mkdir gc
cp /g/solexa/RunVol10/scratch/target/gc/* /tmp/gc/
cd /tmp/gc
ls
```

Now we create two files, one for the gc-values and one for the corresponding coverage values. Afterwards, we use R to plot the average coverage against the gc-content.

```
cut -f 5 targets.gc > gc
awk '{print $5/($3-$2+1)}' NA12878_illumina.cov > ill
```

```
R
```

```
x=scan("gc")
y=scan("ill")
plot(x, log2(y))
y=scan("sol")
plot(x, log2(y))
```

In Figure 5.7(a) and Figure 5.7(b) I created that plot for the Illumina and SOLiD data.

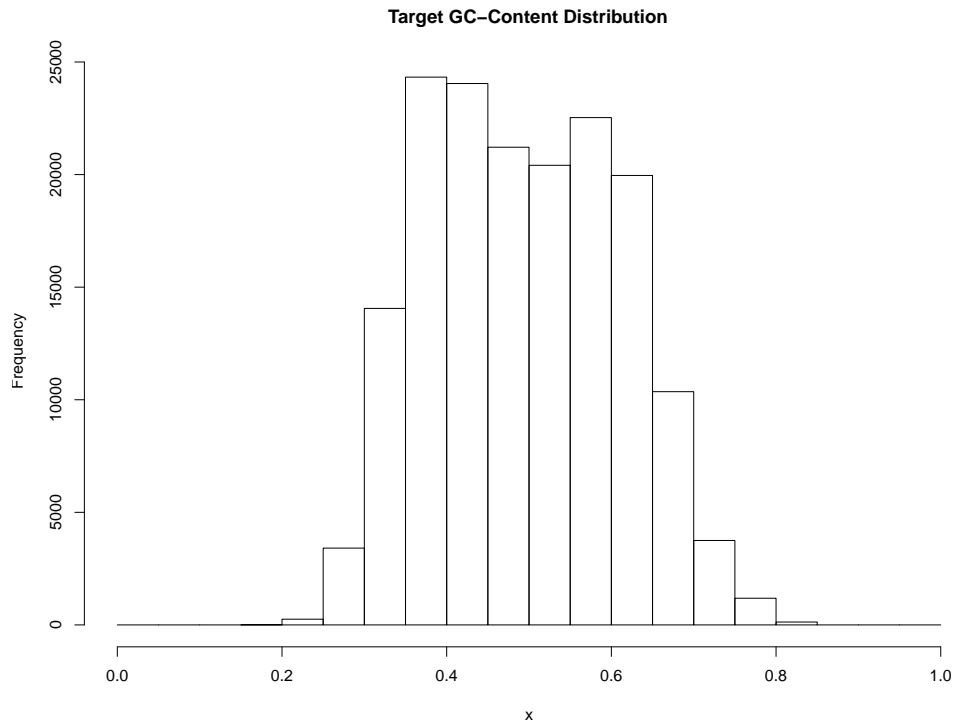
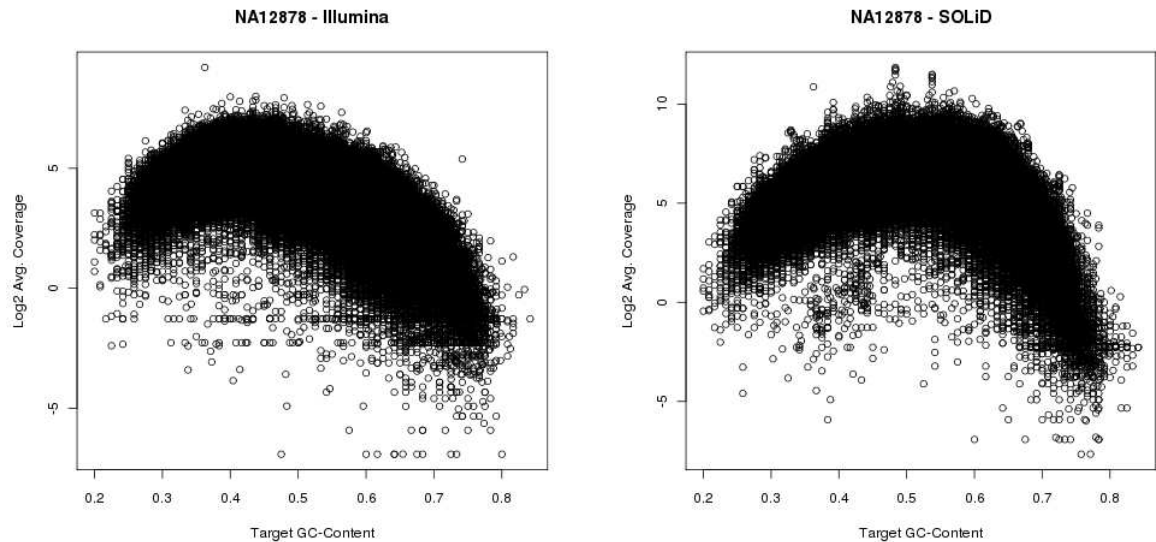


Figure 5.6: Histogram of Target GC-Content



(a) GC Content Illumina

(b) GC Content Solid

Figure 5.7: Avg. Coverage against Target GC-Content

Assembly

6.1 De novo Assembly

De novo assembly is still one of the most computationally demanding problems in Bioinformatics. Because of that we can only consider a very tiny example in this course, which is a BAC assembly of less than a million reads. We will use for that one of the first de Bruijn graph assemblers available, namely Velvet (Zerbino and Birney, 2008).

```
../bin/velvet_1.0.19/velveth out 31 -fastq -shortPaired s_1*_seq.fq
../bin/velvet_1.0.19/velvetg out/ -min_contig_lgth 1000 -cov_cutoff 5
    -ins_length 220 -ins_length_sd 50 -exp_cov 500
```

We actually also have a reference sequence of that BAC which is not exactly identical but of very high identity. To quickly compare the assembled contigs with the original BAC sequence we paste them together in a single FASTA file and visualize the results using MUMmer (Kurtz et al., 2004) or the MAFFT online server (Kato et al., 2002).

```
cat bac.fa > assembly.fa
cat out/contigs.fa >> assembly.fa
```

To see the influence of the k-mer size we can now iterate such an assembly for varying k.

```
#!/bin/bash
for K in 21 23 25 27 29 31
do
    ../bin/velvet_1.0.19/velveth out ${K} -fastq -shortPaired
        s_1*_seq.fq > /dev/null
    STAT='../bin/velvet_1.0.19/velvetg out/ -min_contig_lgth 1000
        -cov_cutoff 5 -ins_length 220 -ins_length_sd 50
        -exp_cov 500 | grep "Final graph"'
    echo ${K} ${STAT}
    rm -rf out
done
```

6.2 Reference-based Assembly

Given a suitable reference sequence we can also apply, so-called reference-based assemblers such as MIRA.

```
cat s_1_1_seq.fq s_1_2_seq.fq > bac_in.solexa.fastq
```

```
cp bac.fa bac_backbone_in.fasta
../bin/mira --project=bac --job=mapping,genome,accurate,solexa --fastq
cat bac_assembly/bac_d_info/bac_info_contigstats.txt
cat bac_assembly/bac_d_info/bac_info_assembly.txt
```

Bibliography

- K. Katoh, K. Misawa, K. Kuma, and T. Miyata. MAFFT: a novel method for rapid multiple sequence alignment based on fast Fourier transform. *Nucleic Acids Res.*, 30: 3059–3066, Jul 2002.
- J. Korbil, A. Abyzov, X. Mu, N. Carriero, P. Cayting, Z. Zhang, M. Snyder, and M. Gerstein. Pomer: a computational framework with simulation-based error models for inferring genomic structural variants from massive paired-end sequencing data. *Genome Biology*, 10(2):R23, 2009.
- S. Kurtz, A. Phillippy, A. L. Delcher, M. Smoot, M. Shumway, C. Antonescu, and S. L. Salzberg. Versatile and open software for comparing large genomes. *Genome Biol.*, 5: R12, 2004.
- H. Li, B. Handsaker, A. Wysoker, T. Fennell, J. Ruan, N. Homer, G. Marth, G. Abecasis, R. Durbin, and . G. P. D. P. Subgroup. The Sequence Alignment/Map format and SAMtools. *Bioinformatics*, 25(16):2078–2079, 2009.
- A. McKenna, M. Hanna, E. Banks, A. Sivachenko, K. Cibulskis, A. Kernytsky, K. Garimella, D. Altshuler, S. Gabriel, M. Daly, and M. A. DePristo. The Genome Analysis Toolkit: A MapReduce framework for analyzing next-generation DNA sequencing data. *Genome Research*, 20(9):1297–1303, 2010.
- T. Rausch. Sequence analysis tools. 2010. URL www.embl.de/~rausch.
- K. Wang, M. Li, and H. Hakonarson. Annovar: functional annotation of genetic variants from high-throughput sequencing data. *Nucleic Acids Research*, 38(16):e164, 2010.
- D. Zerbino and E. Birney. Velvet: Algorithms for de novo short read assembly using de bruijn graphs. *Genome Research*, 2008.

Index

Alignment Statistics, 4
Annovar, 13
Assembly, 26

Bait, 19
BAM, 21

Chromosomal Aberrations, 5
Coverage, 22

De novo Assembly, 26
Detecting Structural Variants, 14
Dotchart, 22

Exons
 Size Distribution, 19
 Total Length, 20

GC-Content, 23
Genome Capture, 19

HapMap, 21

Introduction, 3

Large Deletions, 14

On-target Analysis, 20

Pie chart, 22

Read depth, 4
Reference-based Assembly, 26

SAM, 21
Sample Data, 3
Samtools
 flagstat, 4
 mpileup, 10
samtools, 21
Sequencing Statistics, 21

Short Indel Calling, 10
Single Nucleotide Polymorphisms, 10
SNP Annotation, 10
SNP Calling, 10
Structural Variant, 14

Tandem Duplications, 16
Target Region, 19
Targets
 Size Distribution, 19
 Total Length, 20
 Unmapped targets, 23

UCSC Browser, 19