Genome Sequencing and Structural Variation

Peter Robinson

January 7, 2014

Abstract

This file explains how you can do the analysis discussed in the lecture. Note that the analysis is not necessarily being done in an elegant way or with the most optimal settings, it is merely designed to allow you to get started quickly.

Introduction

The goal of this tutorial is to perform read-depth mapping on human chromosomal data using a simplified version of the algorith described in Yoon S et al. (2009), Sensitive and accurate detection of copy number variants using read depth of coverage, *Genome Res* **19**:1586.–1592.

Data

Download a BAM file from the thousand genomes project data. To keep things simple, I downloaded just two chromosomes from this directory[1]. Here are the files you will need (of course, you can use other files if you like, just change the commands correspondingly).

```
HG00155.chrom11.ILLUMINA.bwa.GBR.low_coverage.20120522.bam
HG00155.chrom11.ILLUMINA.bwa.GBR.low_coverage.20120522.bam.bai
HG00155.chrom20.ILLUMINA.bwa.GBR.low_coverage.20120522.bam
HG00155.chrom20.ILLUMINA.bwa.GBR.low_coverage.20120522.bam.bai
```

Convert the BAM files to their text-based equivalent SAM files. To do this, you can use the samtools package[2]. I am going to show the commands as if everything is visible in PATH.

```
$ samtools view -h -o chrXYZ.sam -q 30 $XYZ.bam
```

You can read the man pages for samtools for a detailed introduction, but the above command converts BAM to SAM format, includes the header (`-h`), filters out reads with a mapping quality below PHRED 30 (`-q 30`), and outputs (`-o`) to a file called chrXYZ.sam. Do this separately for the chromosome 11 and chromsome 20 BAM files. Examine the files now:

```
$ less -S chr11.sam
```

[1] ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/data/HG00154/alignment/
[2] http://samtools.sourceforge.net/

Consult the slides for the first lecture or the SAM documentation for details on the SAM format. You will see there are a number of header lines starting with `@`, followed by lines for individual reads

```
...
...
SRR101476.10329748      99      11      177542  41      76M     =       177892  425     AAATGGAGAA....
...
...
```

You will note that all reads are localized to chromosome 11 (this is the third field). For read depth mapping, we will extract the start positions of the reads from this file (the fourth field, in the above example, 177542). We will use awk to do this. The following command skips lines that start with `@` (i.e., `!/^@/`), and then prints out the fourth field of each line (`print $4`) and redirects the output of awk to a new file called `chr11.pos`.

```
$ awk '!/^@/ {print $4}' chr11.sam > chr11.pos
```

This gives us a file that looks like

```
...
...
7832406
7832433
7832433
7832447
7832465
7832482
...
...
```

We now want to count up the number of reads in 1000 nt windows spread across the chromosome. This is the sort of thing that Perl excels at.

```
#!/usr/bin/perl -w
use strict;
use POSIX;

my %counts;
my $fname = "chr11.pos";
open my $fh,$fname or die "$!";


my $windowsize=1000;
my $c=0;
my $pos = $windowsize;

my $windowsize=10000;
my $c=0;
my $pos = $windowsize;
while (my $x=<$fh>) {
    chomp($x); ## $x is now one of the positions, e.g.,87040297
    my $bucket =  $x - $x%$windowsize + 1000; ## round up to next thousand
```

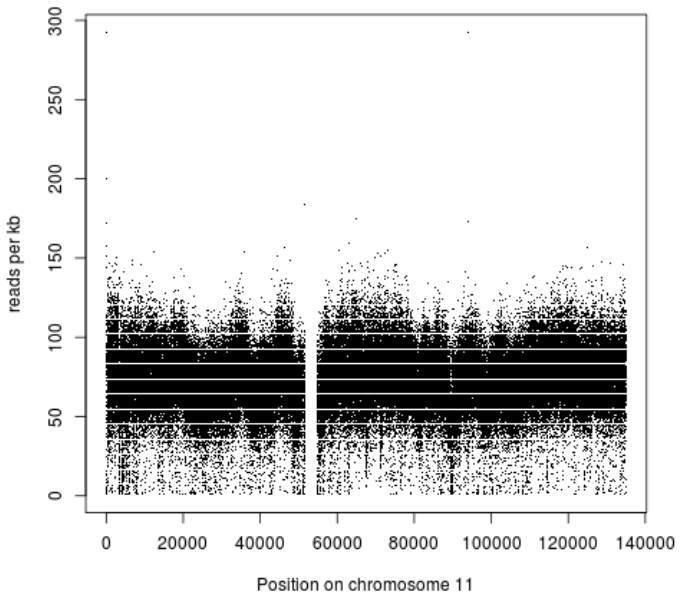
```
    $counts{$bucket}++;
}

## numerical sort
foreach $b (sort { $a <=> $b } keys %counts) {
     print "$b\t$counts{$b}\n";
}
```

Assuming we have the above codde in a file called countReads.pl, we can run the command

```
$ perl countReads.pl > RD.tab
```

Now, we can finally plot the data in R

```
> dat <- read.table("RD.tab", header=FALSE,col.names=c("pos","count"))
> dat$pos <- (1/1e6)*dat$pos  ## correct for window sizee
> plot(dat$pos,dat$count,type="p",pch=".",
         xlab="Position on chromosome 11",ylab="reads per kb")
```



The ggplot2 package in R can be used to make quite nice-looking graphics. or

```
library(ggplot2)
dat <- read.table("RD.tab", header=FALSE,col.names=c("pos","count"))
dat$pos <- (1/1e6)*dat$pos
df <- data.frame(count=dat$count,position=dat$pos)
ggplot(df, aes(x=position, y=count)) +
             geom_point(alpha = .1) +
             theme_bw() +
             xlab("Chr11 (Mb)")
```

GC Bias

Let us now investigate whether there is a GC-bias. The following perl code will get the job done, but this is probably the sort of thing that is better done in C. First, download the file for the human chromosome 11 from UCSC[3] and g-unzip it. Then run the following code.

```
#!/usr/bin/perl -w
use strict;
my $fname="chr11.fa";
open my $fh,$fname or die "$!";
my %gc;
my $ACGT=0;
my $GC=0;
my $windowsize=1000;
my $pos=0;
my $i=0;
while (<$fh>) {
  chomp;
  my @bases = split//;
  ## loop over each base in curent line
  foreach my $b (@bases) {
    $pos++; ## position along chromosome, 1-based
    $b = uc $b; ## upper case
    if ($b eq "C" or $b eq "G") {
      $GC++;
      $ACGT++;
} elsif ($b eq "A" or $b eq "T") {
    $ACGT++;
} ## Note some bases can be "N", just skip them.

if ($pos % $windowsize == 0) { ## we have reached end of window
    if ($ACGT==0) { ## just N bases here
        $gc{$pos}=0;
    } else {
 $gc{$pos}=$GC/$ACGT;
    }
    $GC=0;
    $ACGT=0; ## reset.
}
    }
}

foreach $b (sort { $a <=> $b } keys %gc) {
     print "$b\t$gc{$b}\n";
}
```

We now nead to read both data files into R and to merge them.

```
library(ggplot2)
```

[3] http://hgdownload.cse.ucsc.edu/goldenPath/hg19/chromosomes/.

```
dat <- read.table("RD.tab", header=FALSE,col.names=c("pos","count"))
gc <- read.table("chr11.gc",header=FALSE,col.names=c("pos","gc"))
M <- merge(dat,gc,by="pos")
df <- data.frame(count=M$count,gc=M$gc)
lm1 <- lm(M$count ~ M$gc)
summary(lm1)
## Now plot using ggplot
ggplot(df, aes(x=gc,y=count,color=count)) +
             geom_point(shape=20,alpha=0.1) +
             geom_smooth() +
             theme_bw()
```