Copy Number Analysis for DNA-Seq Data

Oscar M Rueda

August 10, 2014

Contents

1 Introduction

The purpose of this practical is to introduce different methods for analysing copy number data from different types of next-generation sequencing data. Today we will analyze data from whole-genome sequencing with low coverage (shallow sequencing) and whole-exome sequencing.

2 Analysis of whole-genome sequencing data with CNAnorm

2.1 The data

We will use data from a matched tumour/blood sample. Genome has been cut into windows of roughly 426,900 bases long, and the number of reads in each of them has been counted in both the tumour and the blood. The GC content in each window has been computed too.

Use Case: Load the data and inspect it:

```
library(CNAnorm)
data(LS041)
```

```
head(LS041)
```

Use Case: How would we obtain a data.frame like this starting from a bamfile? There are many options. Using the *exomeCopy* we can divide the genome into windows using the function `subdivideGranges`. Then, we would count the reads in the file 'mybamfile.bam' on the windows defined using the function `countBamInGRanges`. We can also get GCcontent usign `getGCcontent` on a reference fasta file. The following code will not work (unless you have the bamfiles 'mybamfile.bam', 'mybamfilenormal.bam and a reference file 'homosapiens.fa')

```
library(exomeCopy)
chr <- paste("chr", c(1:22, "X"), sep="")
start <- 1
end <- c(249.250e06, 243.194e06, 198.022e06, 191.154e06, 180.915e06,
         171.115e06, 159.138e06, 146.364e06, 141.213e06, 135.534e06,
         135.006e06, 133.851e06, 115.169e06, 107.349e06, 102.531e06,
         90.354e06, 81.195e06, 78.077e06, 59.128e06, 63.025e06,
         48129895, 51.304e06, 155.270e06)
gr <- GRanges(seqname=chr, IRanges(start=start, end=end))
gr <- subdivideGRanges(gr, 100000)
Reads <- RangedData(space=seqnames(gr), ranges=ranges(gr))
Reads[['Sample1']] <- countBamInGRanges("mybamfile.bam", gr)
Reads[['Normal']] <- countBamInGRanges("mybamfilenormal.bam", gr)
Reads[["GC"]] <- getGCcontent(gr, "homosapiens.fa")
Reads <- as.data.frame(Reads)
```

2.2 Nomalization and copy number estimation

Use Case: Coming back to our practical and our dataset, now we convert the data into CNAnorm object. As we can see, the first thing we do is to compute the ratio of tumour/normal.

```
CN <- dataFrame2object(LS041)
CN
summary(ratio(CN))
```

Use Case: The first normalizing step is GC-correction: GC content can produce sequencing bias; therefore we apply a loess correction. Note that we exclude chromosomes Y and M

```
toSkip <- c("chrY", "chrM")
CN <- gcNorm(CN, exclude=toSkip)
```

Use Case: The next step is to smooth the GC-corrected ratios to increase the signal to noise:

```
CN <- addSmooth(CN)
CN
boxplot(log2(ratio(CN)), log2(ratio.s(CN)), names=c("Ratios", "Smoothed ratios"),
ylab="log-ratios")
```

Use Case: Now we try to determine the ploidy and tumour content of the sample using the `peakPloidy`. There are several available methods for this task. The default is "mixture", based on a mixture model but it assumes that the sample is mostly monoclonal. "closest" is a simpler methods that works under more general assumptions, but does not estimate ploidy and cellularity.

```
par(mfrow=c(1,2))
CN <- peakPloidy(CN, exclude=toSkip, method="closest")
plotPeaks(CN)
CN <- peakPloidy(CN, exclude=toSkip, method="mixture")
plotPeaks(CN)
```

Use Case: Now we will validate the peaks, segment the ratios using the *DNAcopy* CBS method, so we can obtain genomic regions that share the same copy number and finally normalize the data using the plody and tumour content information.

```
CN <- validation(CN)
CN <- addDNACopy(CN)
CN <- discreteNorm(CN)
CN
```

Use Case: Let's plot the data:

```
par(mfrow=c(1,1))
plotGenome(CN, superimpose="DNACopy", show.centromeres=FALSE)
plotGenome(CN[which(chrs(CN)=="chr8")], superimpose="DNACopy",
show.centromeres=FALSE)
```

Use Case: Finally, we can export the values into a text file. We can export ratios copy numbers.

```
exportTable(CN, "copyNumbers.txt", show="ratio")
```

See the help file for other options to show.

3 Analysis of exome sequencing data with exomeCopy

Target enrichment usually produces different in read depths dependent on the target and not the sample, therefore one needs to use a target specific normalization to remove this dependence. Please note that *exomeCopy* is not recommended for tumour samples, but for CNV detection in normal samples.

3.1 The data

We will use data from a small subset of chromosome 1 frmo the 1,000 Genomes Project.

Use Case: Read the data and get the counts

```
library(exomeCopy)
data(exomecounts)
head(exomecounts)
plot(start(exomecounts), exomecounts$HG00551, xlim = c(8e+05, 1800000),
xlab = "genomic position", ylab = "counts",
main = "HG00551 read counts in exonic ranges", pch=19)
```

Note that you can use the function `countBamInGRanges` described before with your bam files andthe targets of your experiment to obtain an object like this. GC-content can be computed with `getGCcontent`.

3.2 Normalization and copy number estimation

Use Case: First, we need to obtain the background read depth, that is an estimate of the normal copy number per target and other measures that might be useful for normalizing our data.

```
pool.samples <- colnames(exomecounts)[grep("HG", colnames(exomecounts))]
exomecounts[["bg"]] <- generateBackground(pool.samples, exomecounts, median)
exomecounts[["log.bg"]] <- log(exomecounts[["bg"]] + 0.1)
exomecounts[["bg.var"]] <- generateBackground(pool.samples, exomecounts, var)
exomecounts <- exomecounts[exomecounts[['bg']]>0,]
exomecounts[["GC.sq"]] <- exomecounts[["GC"]]^2
exomecounts[['width']] <- width(exomecounts)
```

Note that we have removed exomes with zero background coverage, as these are probably difficult to enrich.

Use Case: *exomeCopy* fits a Hidden Markov model on the reads based on a Poisson distribution and several covariates named "positional effects'.

```
fit <- exomeCopy(exomecounts, sample.name=pool.samples[2],
X.names=c("log.bg", "GC", "GC.sq", "width"), S=0:6, d=2)
show(fit)
```

Note that the function is applied individually to each chromosome and sample. Note that we set a maximum of 6 copies and we expect the sample to be diplod (arguments "S" and "d").

Use Case: Let us inspect the results

```
copyCountSegments(fit)
plot(fit)
```

Use Case: We need to use a somewhat intrincated way of analyzing all samples and put them together in a convenient object for later plotting

```
fit.list <- lapply(pool.samples, function(sample.name) {
    lapply(seqlevels(exomecounts), function(seq.name) {
        exomeCopy(exomecounts[seq.name], sample.name,
X.names = c("log.bg", "GC", "GC.sq", "width"), S = 0:4, d = 2)
    })
})

tmp <- list()
for (i in 1:length(fit.list)) {
tmp[[i]] <- do.call("rbind", lapply(fit.list[[i]], copyCountSegments))
}
all.fits <- do.call("rbind", tmp)
all.fits
cnv.cols <- c("red", "orange", "black", "deepskyblue","blue")
plotCompiledCNV(all.fits, seq.name="chr1", col=cnv.cols)
```