ABCD-DNA: Copy-number-aware differential analysis of quantitative DNA sequencing data

Mark D. Robinson

Last compiled on: July 27, 2012

1 Introduction

This document runs through all the steps and gives some extra statistical details regarding copy-number-aware differential analyses of *quantitative* sequencing data. We are under the assumption that copy number variation (CNV) data is available on the same DNA that has been profiled through some quantitative DNA sequencing assay (e.g. ChIP-seq, MBDCap-seq; we refer to these collectively as QDNA-seq).

The general idea of ABCD-DNA (**A**ffinity-**B**ased **C**opy-number-aware **D**ifferential analysis of quantitative **DNA** sequencing) is that higher (lower) copy number results in higher (lower) read density, assuming everything else equal (e.g. methylation levels for MBDCap-seq, or enrichment strength for ChIP-seq assays); this is not new or unexpected. However, the challenge comes in how to integrate this knowledge into a statistical analysis that is searching for differential regions of enrichment. Our goal here is to disengtangle changes in the epigenome from changes in the epigenome. In addition, there are few methods available to do *differential* analyses between ChIP-seq sample that allow replication, careful normalization and so on; many approaches are being actively researched. Borrowing and adapting the analyses and infrastructure used in RNA-seq, ABCD-DNA addresses both of these challenges. We propose to integrate the CNV information directly as offsets in a count-based statistical model. An annotated analysis for a comparison using MBDCap-seq (methylated DNA capture) between LNCaP (prostate cancer) and PrEC (prostate epithelial) cells is given below.

Further Supplementary information, including R scripts and pre-processed data, can be found at:
`http://imlspenticton.uzh.ch/robinson_lab/ABCD-DNA/`

In this document, we go through the details of how the ABCD-DNA analysis was designed. In Section 5, we discuss the interface that is now available in the Bioconductor ***Repitools*** package.

2 Load R package and data

Regardless of QDNA-seq dataset, the starting point is a table of counts for all regions of interest, across all samples of interest, with corresponding CNV information for each sample. Here, we load a preprocessed dataset comprising one of the datasets used in the main paper.

Our data is available in unprocessed form from the Gene Expression Omnibus, or in processed form from the Supplementary website[1] as R objects with read densities and CNV across 500 base

[1] `http://imlspenticton.uzh.ch/robinson_lab/ABCD-DNA/`

pair bins genome-wide.

First, we load the necessary packages and the pre-computed R objects:

```
> library(GenomicRanges)
> library(edgeR)
> load("../Rdata_created/counts.Rdata")
> load("../Rdata_created/orig_regs_gb.Rdata")
```

In this example, the count table consists of technical replicates of MBDCap-seq for LNCaP, PrEC and SssI (fully methylated control) DNA:

```
> head(counts)
```

```
  LNCaP_MBD2IP_1 LNCaP_MBD2IP_2 PrEC_MBD2IP_1 PrEC_MBD2IP_2
1              2              0             0             1
2              1              0             0             1
3              0              0             0             0
4              0              0             0             0
5              0              0             0             0
6              0              0             0             0
  SssI_Elution_6_MBD2IP_1 SssI_Elution_6_MBD2IP_2
1                       6                       4
2                       6                       3
3                       0                       0
4                       2                       3
5                       3                       3
6                       0                       0
```

The SssI control is only used here to pre-select the regions of the genome (see below), since it reflects where the assay actually enriches for methylated DNA. This step is not necessary for other assays, but some subsetting is recommended if nothing else to reduce the burden of multiple testing corrections.

To keep track of the regions of the genome for the count table, we have a corresponding `GRanges` object, as follows:

```
> gb
```

```
GRanges with 5570057 ranges and 0 elementMetadata cols:
            seqnames               ranges strand
               <Rle>            <IRanges>  <Rle>
        [1]     chr1       [51501, 52000]      *
        [2]     chr1       [52001, 52500]      *
        [3]     chr1       [52501, 53000]      *
        [4]     chr1       [53001, 53500]      *
        [5]     chr1       [53501, 54000]      *
        [6]     chr1       [54001, 54500]      *
        [7]     chr1       [54501, 55000]      *
        [8]     chr1       [55001, 55500]      *
        [9]     chr1       [55501, 56000]      *
        ...      ...                  ...    ...
  [5570049]     chrY [57708501, 57709000]      *
```

```
  [5570050]     chrY [57709001, 57709500]      *
  [5570051]     chrY [57709501, 57710000]      *
  [5570052]     chrY [57710001, 57710500]      *
  [5570053]     chrY [57710501, 57711000]      *
  [5570054]     chrY [57711001, 57711500]      *
  [5570055]     chrY [57711501, 57712000]      *
  [5570056]     chrY [57712001, 57712500]      *
  [5570057]     chrY [57712501, 57713000]      *
  ---
  seqlengths:
    chr1  chr2  chr3  chr4  chr5 ... chr20 chr21 chr22  chrX  chrY
      NA    NA    NA    NA    NA ...    NA    NA    NA    NA    NA
```

As you can see, these are 500bp bins tiled along the genome.

These were pre-computed using the `annotationBlocksCounts` function in the ***Repitools***[2] package. As an alternative starting point, if you have a `GRangesList` of mapped reads or a set of BAM files, you can use `annotationBlocksCounts` to create a table of read densities. In addition, there are plenty of tools available to count read densities across genomic regions, such as ***HTSeq-count***[3], ***Rsubread***[4] and so on.

In our case, the CNV information originates from Affymetrix SNP 6.0 arrays and can be pre-processed into a matrix, one each for LNCaP and PrEC cells, and that corresponds to the counts table. For simplicity, we are only considering regions of the normal PrEC genome with 2 copies, but this can be easily be extended.

```
> table(regs)
```

```
regs
L=1 P=2 L=2 P=2 L=3 P=2 L=4 P=2 L=5 P=2
  16943  273705 1207077 3800068  272264
```

```
> length(regs)
```

```
[1] 5570057
```

```
> length(regs)==length(gb)
```

```
[1] TRUE
```

CNV information (e.g. genotyping arrays, genome sequencing) is segmented and software will output the genomic ranges and their segmental copy number values. For our analyses, we used the PICNIC tool[5], parsed the segmented copy numbers into a `GRanges` object and matched it to our regions of interest (code not shown; see 'preprocess.R' at the Supplemental web site). Insert your tool of choice here and parse the segmented copy numbers into a `GRanges` object, which can be easily matched to the GRanges object of regions of interest using the `findOverlaps` function in the `IRanges` package.

[2] http://bioinformatics.oxfordjournals.org/content/26/13/1662.abstract
[3] http://www-huber.embl.de/users/anders/HTSeq/doc/index.html
[4] http://www.bioconductor.org/packages/release/bioc/html/Rsubread.html
[5] http://www.sanger.ac.uk/genetics/CGP/Software/PICNIC/

3 Normalization

Normalization is here accomplished in two parts: i) sample-specific adjustments for depth and composition; ii) adjustments for copy number, which come from the pre-processed CNV information. These two components need to be integrated together.

3.1 Sample-specific factors

We first take a subset of the data to make it manageable, using a fairly liberal cutoff of regions that can be reasonably interrogated with MBD-seq:

```
> rs <- rowSums(counts[,5:6])  # read depth in SssI
> k <- rs>10
> counts <- counts[,1:4]
> grp <- factor( gsub("_[12]","",colnames(counts)), levels=c("PrEC_MBD2IP","LNCaP_MBD2IP"))
> counts <- counts[k,]
> gb <- gb[k,]
> regs <- regs[k]
> dim(counts)
```

```
[1] 793838      4
```

All of the count-based modeling takes place in the edgeR package, which can accept region- and sample-specific offsets. That is, ABCD-DNA is a procedure for specifying this table of offsets, where each cell of the matrix corresponds with the count matrix. If a given genomic region is amplified, it should have a higher value of its offset. Similarly, if a sample is sequenced to a higher depth, it will also attract a higher offset; of course, the depth offset has the subtlety that composition of the capture DNA population will have an effect on the expect read count. Overall, this framework allows full flexibility for different regions and different samples to have different copy number states. In our example, copy-number offsets are estimated externally using Affymetrix SNP 6.0, we check that these in fact have a roughly linear impact on the read depth in the QDNA-seq (aside from the biological changes that occur) and the relative depth offsets are estimated from the "neutral", or most prominent state. These offsets are delivered to the generalized linear model to adjust the statistical testing. The advantage of this strategy is that the read densities are kept on their original scale, which allows straightforward mean-variance modeling.

To illustrate this analysis, we start with creating a standard `DGEList` object:

```
> # calculate standard p-values / CN-aware p-values
> # dataset-specific
> d <- DGEList(counts=counts,group=grp,genes=as.data.frame(gb)[,-c(4:5)])
> d <- calcNormFactors(d)
> d
```

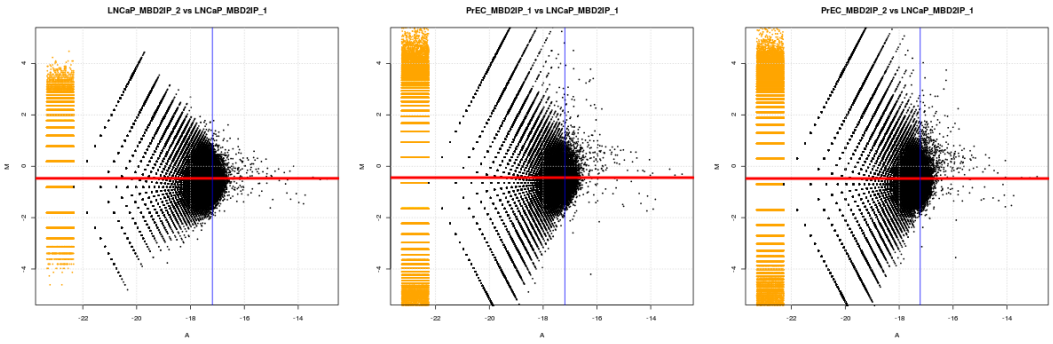
```
An object of class "DGEList"
$samples
                     group lib.size norm.factors
LNCaP_MBD2IP_1 LNCaP_MBD2IP  5205141    1.0061212
LNCaP_MBD2IP_2 LNCaP_MBD2IP  9180957    1.0279797
PrEC_MBD2IP_1   PrEC_MBD2IP  8721579    0.9841434
```

```
PrEC_MBD2IP_2   PrEC_MBD2IP  9037020    0.9824417

$counts
    LNCaP_MBD2IP_1 LNCaP_MBD2IP_2 PrEC_MBD2IP_1 PrEC_MBD2IP_2
54               8             10             6             6
55               9             10             7             5
150              2              2             3             2
946              3              7             4             1
970             17             19             7             9
793833 more rows ...

$genes
    seqnames  start    end
54      chr1  78001  78500
55      chr1  78501  79000
150     chr1 126001 126500
946     chr1 524001 524500
970     chr1 536001 536500
793833 more rows ...
```

The call to `calcNormFactors` is simply to populate various variables in the object; these will be modified to give the sample-specific normalization factors.

As mentioned in the main paper, LNCaP is predominantly 4 copies and PrEC is a normal genome generally with 2 copies. Thus, we treat this "L=4 P=2" state as a reference point and estimate all the sample-specific factors for this subset of the genome. We calculate these normalization factors using the regions of lower variability (i.e. higher counts where log-ratios are more precisely estimated) for just the "neutral" state. We plot them graphically as M (log-ratio of depth-adjusted count) versus A (log-average depth-adjusted counts) plots:

```
> ref <- 1
> nf <- rep(0,ncol(d))
> par(mfrow=c(1,3))
> for(i in 2:ncol(counts)) {
+   tit <- paste(colnames(d)[i], "vs", colnames(d)[ref])
+   map <- maPlot(d$counts[regs=="L=4 P=2",ref], d$counts[regs=="L=4 P=2",i],
+                 normalize=TRUE, pch=19, cex=.3, ylim=c(-5,5), main=tit); grid();
+   q <- quantile(map$A[!map$w],.99)
+   nf[i] <- median( map$M[map$A > q] )
+   abline(h=nf[i],col="red",lwd=4)
+   abline(v=q,col="blue")
+ }
> z <- exp(nf)
> z <- z/(prod(z)^(1/length(z)))
> d$samples$norm.factors <- z
```

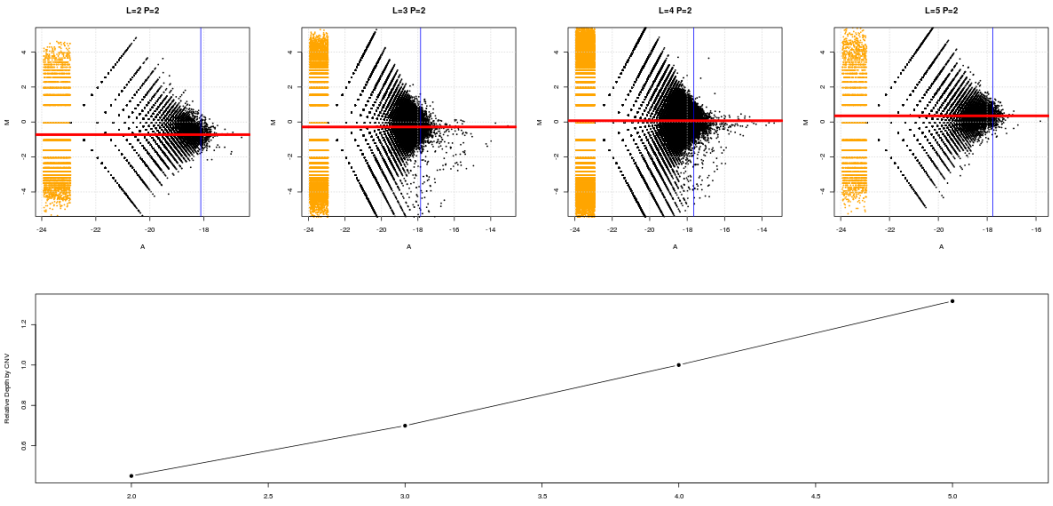


The `norm.factors` vector in the `$samples` element now carries the offsets corresponding to "effective" read depth, which makes the prominent copy number state (i.e. L=4 P=2) the reference state.

3.2 Region-specific copy-number factors

Now, we need to introduce the copy number offsets. Our assumption is that a doubling of copy number should result in a double of the average read depth, or more generally that there is a linear relationship between copy number estimates and read depth in the QDNA-seq assays. This is, of course, only an approximation, since many regions may change in their enrichment levels. In addition, depending on your platform used for copy number detection, there may be attenuation issues (i.e. non-linearities). In addition, many microarray-based segmentation algorithms operate on the log scale; these should be exponentiated to the linear scale for introduction through ABCD-DNA. In the main manuscript, we demonstrated an approximate linear relationship across 3 distinct QDNA-seq datasets for the CNV-typed cell lines LNCaP and PrEC. We repeat this exercise here to illustrate the mechanics on 1 of the LNCaP and PrEC replicates.

```
> layout(matrix(c(1:4,rep(5,4)),2,4,byrow=TRUE))
> cn <- c("L=2 P=2","L=3 P=2","L=4 P=2","L=5 P=2")
> f.by.cn <- rep(NA,length(cn))
> els <- d$samples$lib.size*d$samples$norm.factors # effective library size
> ref <- 3
> sam <- 2
> for(i in 1:length(cn)) {
+   kk <- regs==cn[i]
+   map <- maPlot(d$counts[kk,ref]/els[ref], d$counts[kk,sam]/els[sam],
+               normalize=FALSE, pch=19, cex=.3, ylim=c(-5,5)); grid();
+   top <- max(100,round(.01*sum(!map$w)))
+   o <- order(-map$A)[1:top]
+   q <- max(min(map$A[o]),max(map$A[map$w]))
+   abline(v=q,col="blue");
+   f.by.cn[i] <- median(map$M[map$A>q])
+   abline(h=f.by.cn[i],col="red",lwd=4);
+   title(cn[i])
+ }
> f <- exp(f.by.cn)
> plot(2:5,f/f[3],pch=19,ylab="Relative Depth by CNV",type="b",
+      xlab="",xaxs="i",xlim=c(1.65,5.35))
```



We would recommend an analysis like to this to confirm that changes in copy number are reflected in changes in QDNA-seq data before continuing. If there is no obvious linear (or other) relationship, then it points to confirming other aspects of the experiment – is the same source of DNA used for CNV-typing and QDNA-seq? are the CNV estimates on the right scale? did the ChIP/MBD enrichment or sequencing work properly? And, so on.

Of course, in many real-world applications, the segmented copy number state may be non-integer (e.g. mixtures of cell populations). Before applying these checks, modifications to the above integer-based plot will need to be made (e.g. grouping into copy number groups). Regardless, an association between DNA copy and QDNA-seq density should be established before proceeding.

In any case, we can now create a matrix of offsets for our dataset. Because we have normalized 4 copies for LNCaP to 2 copies for PrEC, we are basically treating these as neutral (and that there relative read densities don't deviate too much on the average – of course, there will be biological differences). Therefore, we are effectively treating 4 copies in LNCaP as *on par* with 2 copies in PrEC; we need to specify this in our

```
> m <- matrix(rep(c(1,0,1),c(2,4,2)),nrow=2,byrow=TRUE)
> cnL <- as.numeric(gsub("L=","",gsub(" P=2","",regs)))
> cn <- cbind(cnL/2,2) %*% m
> cn[34901:34910,]
```

```
      [,1] [,2] [,3] [,4]
 [1,]  2.0  2.0    2    2
 [2,]  2.0  2.0    2    2
 [3,]  2.0  2.0    2    2
 [4,]  2.5  2.5    2    2
 [5,]  2.5  2.5    2    2
 [6,]  2.5  2.5    2    2
 [7,]  2.5  2.5    2    2
 [8,]  2.5  2.5    2    2
 [9,]  2.5  2.5    2    2
[10,]  2.5  2.5    2    2
```

Here, the `m` matrix is just to expand the single columns of CNV estimates to the 2 columns each (technical replicates) for the read densities.

4 Generalized Linear Model fitting with offsets

Next, we introduce generalized linear modeling. The commands below illustrate how you might do a standard differential analysis on a QDNA-seq data table using ***edgeR***:

```
> # Fit the NB GLMs
> design <- model.matrix(~grp)
> rownames(design) <- colnames(d)
> design
```

```
                (Intercept) grpLNCaP_MBD2IP
LNCaP_MBD2IP_1           1                1
LNCaP_MBD2IP_2           1                1
PrEC_MBD2IP_1            1                0
PrEC_MBD2IP_2            1                0
attr(,"assign")
[1] 0 1
attr(,"contrasts")
attr(,"contrasts")$grp
[1] "contr.treatment"
```

```
> fit <- glmFit(d, design, dispersion=0.05)
> lrt <- glmLRT(fit, coef=2)
```

First, we define a design matrix `design`, which in this case just represents a simple 2-group comparison; more complicated designs are supported. We can fit models for each gene using edgeR's `glmFit` and do a likelihood ratio test using `glmLRT` (`coef=2` for changes between LNCaP and PrEC MBD-seq given the design matrix above). For simplicity, we have hard-coded the dispersion to be .05, since we do not have proper biological replicates. However, these are cell lines, so we expect a low baseline level of variation. See the edgeR user's guide[6] and the GLM manuscript[7] for further details.

Next, we need to modify the above analysis to deliver a matching matrix of offsets to go along with the matrix of counts. From the `cn` matrix we created before and the effective library sizes we calculated, it is easy to put these two pieces together, as follows:

```
> o <- outer( rep(1,nrow(d)), getOffset(d)) + log(cn)
> dim(o)
```

```
[1] 793838      4
```

```
> dim(d)
```

```
[1] 793838      4
```

[6] http://www.bioconductor.org/packages/release/bioc/html/edgeR.html
[7] http://www.ncbi.nlm.nih.gov/pubmed/22287627

```
> fitadj <- glmFit(d, design, offset=o, dispersion=.05)
> lrtadj <- glmLRT(fitadj, coef=2)
```

As above, we follow the steps above after introducing the offsets. We can use the facilities within edgeR to come up with CNV-aware lists of differentials regions.

```
> topTags(lrtadj)
```

```
Coefficient:  grpLNCaP_MBD2IP
        seqnames     start       end     logFC   logCPM       LR
5387202    chr20  46565501  46566000 -7.913104 2.287681 136.0044
675197      chr2  90960501  90961000 -7.752314 2.537993 127.7426
5387201    chr20  46565001  46565500 -7.668903 2.044691 124.9497
3429276    chr10  41678001  41678500 -6.478397 2.240175 122.1347
5385985    chr20  45957001  45957500 -7.571460 1.947775 120.6371
4769509    chr16  44943501  44944000 -5.104627 3.350697 119.0972
282727      chr1 141480001 141480500 -5.296381 2.729242 118.4303
5569397     chrY  57382501  57383000 -9.844084 2.118213 118.0796
282726      chr1 141479501 141480000 -5.073060 2.878753 117.2169
1471934     chr4  48817501  48818000 -5.665856 2.382306 115.5838
                                          PValue
5387202 0.00000000000000000000000000001990896
675197  0.00000000000000000000000000127787138
5387201 0.00000000000000000000000000522009087
3429276 0.00000000000000000000000002156840756
5385985 0.00000000000000000000000004588261931
4769509 0.00000000000000000000000009972101236
282727  0.00000000000000000000000013957342559
5569397 0.00000000000000000000000016656539303
282726  0.00000000000000000000000025733365646
1471934 0.00000000000000000000000058628542790
                                        FDR
5387202 0.0000000000000000000001580449
675197  0.0000000000000000000050721143
5387201 0.0000000000000000000138130216
3429276 0.0000000000000000000428045538
5385985 0.0000000000000000000728467335
4769509 0.0000000000000000001319372150
282727  0.0000000000000000001582838415
5569397 0.0000000000000000001652824231
282726  0.0000000000000000002269791502
1471934 0.0000000000000000004654156515
```

Further downstream analyses could then be applied.

5 Using the wrapper software in the *Repitools* package

The above R code and description illustrated the "guts" of how the ABCD-DNA analysis was designed and invoked. In this Section, we describe the software available in the ***Repitools*** Bioconductor package for doing these same steps.

First, we load the package and collect all the information needed: the count table, GRanges object of regions, a design matrix, the copy number calls, and the set of regions deemed to be from the neutral state. From these, we can construct a QdnaData object, as follows:

```
> library(Repitools)
> qd <- QdnaData(counts=counts, regions=gb, design=design,
+                cnv.offsets=cn, neutral=(regs=="L=4 P=2"))
```

As above, the first step is to calculate the sample-specific factors from the neutral state:

```
> qd <- getSampleOffsets(qd,ref=1)
> qd$DGEList$samples
```

```
               group lib.size norm.factors
LNCaP_MBD2IP_1     1  5205141    1.4151518
LNCaP_MBD2IP_2     1  9180957    0.8849364
PrEC_MBD2IP_1      1  8721579    0.9094978
PrEC_MBD2IP_2      1  9037020    0.8779772
```

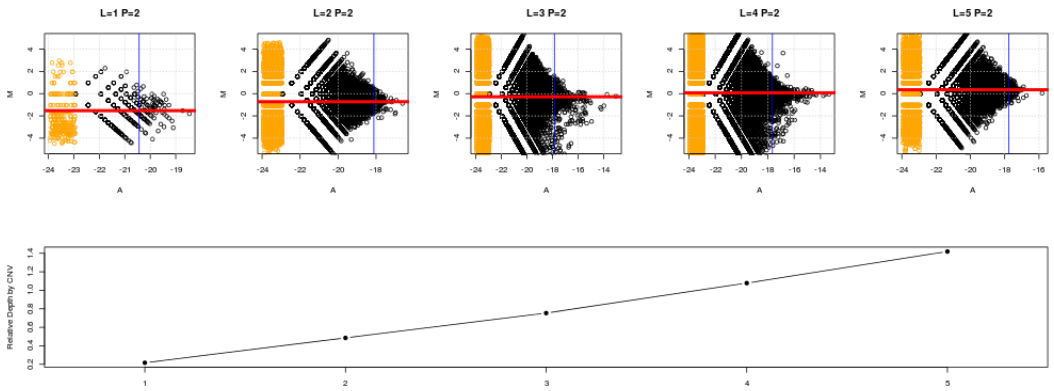
The normalization factors are then stored in the $DGEList$samples element of the QdnaData object. Note that you can use a different reference sample, and it has only a small impact on the normalization factors.

```
> print(getSampleOffsets(qd,ref=3,force=TRUE)$DGEList$samples)
```

```
               group lib.size norm.factors
LNCaP_MBD2IP_1     1  5205141    1.4203819
LNCaP_MBD2IP_2     1  9180957    0.8657193
PrEC_MBD2IP_1      1  8721579    0.9128591
PrEC_MBD2IP_2      1  9037020    0.8908694
```

As before, we can create a plot of the relative change in qDNA-seq across some pre-defined CNV states. Note here that we need to give indices of two samples to compare: 3-PrEC, 2-LNCaP.

```
> plotQdnaByCN(qd,cnv.group=regs,idx.ref=3,idx.sam=2)
```



In the case that the scale of the CNV offsets needs to be changed after inspection of the above plot, there is an accessory function named `setCNVOffsets` to attach CNV offsets to the `QdnaData` object.

Lastly, when we are happy that the offsets reflect a change in read density, we can combine the sample-specific offsets with the CNV offsets, fit the generalized linear model and calculate a likelihood ratio test for the difference of interest. This can be accomplished in a single step, as follows:

```
> f <- abcdDNA(qd, dispersion=.05, coef=2)
> topTags(f)

Coefficient:  grpLNCaP_MBD2IP
        seqnames     start       end     logFC   logCPM       LR
5387202    chr20  46565501  46566000 -7.913104 2.287681 136.0044
675197      chr2  90960501  90961000 -7.752314 2.537993 127.7426
5387201    chr20  46565001  46565500 -7.668903 2.044691 124.9497
3429276    chr10  41678001  41678500 -6.478397 2.240175 122.1347
5385985    chr20  45957001  45957500 -7.571460 1.947775 120.6371
4769509    chr16  44943501  44944000 -5.104627 3.350697 119.0972
282727      chr1 141480001 141480500 -5.296381 2.729242 118.4303
5569397     chrY  57382501  57383000 -9.844084 2.118213 118.0796
282726      chr1 141479501 141480000 -5.073060 2.878753 117.2169
1471934     chr4  48817501  48818000 -5.665856 2.382306 115.5838
                                          PValue
5387202 0.00000000000000000000000000001990896
675197  0.00000000000000000000000000127787138
5387201 0.00000000000000000000000000522009087
3429276 0.00000000000000000000000002156840756
5385985 0.00000000000000000000000004588261931
4769509 0.00000000000000000000000009972101236
282727  0.00000000000000000000000013957342559
5569397 0.00000000000000000000000016656539303
282726  0.00000000000000000000000025733365646
1471934 0.00000000000000000000000058628542790
                                        FDR
5387202 0.0000000000000000000000001580449
675197  0.0000000000000000000000050721143
5387201 0.0000000000000000000000138130216
3429276 0.0000000000000000000000428045538
5385985 0.0000000000000000000000728467335
4769509 0.0000000000000000000001319372150
282727  0.0000000000000000000001582838415
5569397 0.0000000000000000000001652824231
282726  0.0000000000000000000002269791502
1471934 0.0000000000000000000004654156515
```

This is the same table as before (i.e. `f` is the likelihood-ratio test output).

6 Environment

This document was created using the following R version and associated packages:

```
> sessionInfo()


R Under development (unstable) (2012-06-14 r59564)
Platform: x86_64-unknown-linux-gnu (64-bit)

locale:
 [1] LC_CTYPE=en_CA.UTF-8       LC_NUMERIC=C
 [3] LC_TIME=en_CA.UTF-8        LC_COLLATE=en_CA.UTF-8
 [5] LC_MONETARY=en_CA.UTF-8    LC_MESSAGES=en_CA.UTF-8
 [7] LC_PAPER=C                 LC_NAME=C
 [9] LC_ADDRESS=C               LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_CA.UTF-8 LC_IDENTIFICATION=C

attached base packages:
[1] grid      stats     graphics  grDevices utils     datasets
[7] methods   base

other attached packages:
 [1] ROCR_1.0-4
 [2] gplots_2.11.0
 [3] MASS_7.3-19
 [4] KernSmooth_2.23-8
 [5] caTools_1.13
 [6] bitops_1.0-4.1
 [7] gdata_2.11.0
 [8] gtools_2.7.0
 [9] VennDiagram_1.2.1
[10] lattice_0.20-6
[11] rtracklayer_1.17.11
[12] edgeR_2.99.2
[13] limma_3.13.8
[14] BSgenome.Hsapiens.UCSC.hg18_1.3.17
[15] BSgenome_1.25.3
[16] Biostrings_2.25.6
[17] Repitools_1.3.3
[18] GenomicRanges_1.9.29
[19] IRanges_1.15.17
[20] BiocGenerics_0.3.0

loaded via a namespace (and not attached):
[1] RCurl_1.91-1     Rsamtools_1.9.19 stats4_2.16.0
[4] tools_2.16.0     XML_3.9-4        zlibbioc_1.3.0
```