CNV detection in exome sequencing data using exomeCopy

Michael Love
love@molgen.mpg.de

March 19, 2012

Abstract

exomeCopy is an R package implementing a hidden Markov model[1] for predicting copy number variants (CNVs) from exome/targeted sequencing experiments without paired control experiments as in tumor/normal sequencing.[2] It models read counts in genomic ranges using negative binomial emission distributions depending on a hidden state of the copy number and on positional covariates such as GC-content and background read depth. Normalization and segmentation are performed simultaneously, eliminating the need for preprocessing of the raw read counts.

Contents

1 Introduction

The exomeCopy package was designed to address the following situation:

[1]The manuscript describing the model [Love et al., 2011] is listed in the references of this vignette.
[2]R packages which can be used for CNV detection in whole genome or tumor/normal paired exome sequencing data are ReadDepth and ExomeCNV respectively, available on CRAN.

- Target enrichment, such as exome enrichment, leads to non-uniform read depth, which is often correlated across samples.
- CNVs overlapping enriched regions can be detected as increases or decreases in read counts relative to "background" read depth, generated by averaging over a control set.
- Individual samples can be more or less correlated with background read depth and have different dependencies on GC-content.

While exome sequencing is not designed for CNV genotyping, it can nevertheless be used for finding CNVs which overlap exons and are not common in the control set. It can provide an independent data source to be used in combination with higher resolution array-based methods. In this vignette we show how to prepare data, generate background read depth, simulate CNVs in read count data and recover CNVs using exomeCopy . We build a wrapper function for calling the `exomeCopy` function over multiple chromosomes and samples such that the operations can be assigned across workstations.

2 Importing experiment data

The necessary genomic range information, read counts and positional covariates (background read depth and GC-content) should be stored in a *RangedData* object. The user must provide genomic ranges of targeted enrichment. For exome sequencing, one can use exon annotations, which will be discussed in a later section. The exomeCopy package provides two convenience functions for preparing the genomic ranges and sample read counts: `subdivideGRanges` and `countBamInGRanges`. In this vignette we will use a prepackaged *RangedData* object containing real exome sequencing read counts. Due to memory constraints, we cannot construct this object from scratch, but below we demonstrate with a simple example how to construct a *RangedData* object from a BED file describing the targeted region, BAM files for the read mapping and a FASTA file for the reference sequence.

2.1 Subdividing targeted regions

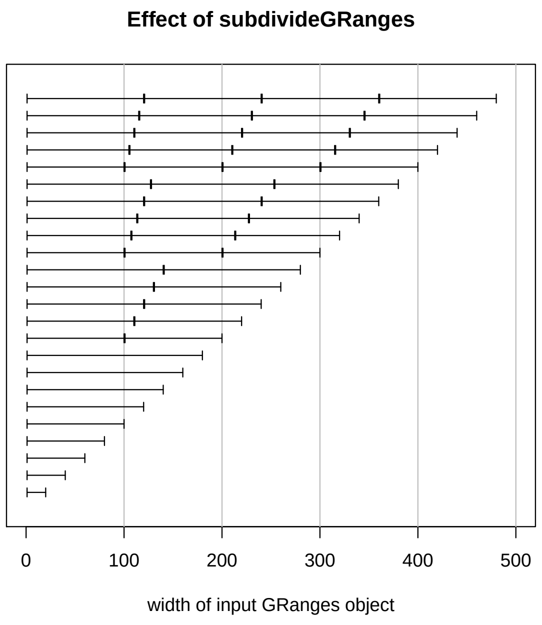
`subdivideGRanges` divides the targeted genomic ranges into a set of ranges of nearly equal width, which exactly cover the original ranges. While exomeCopy can use range width as a covariate for modeling read counts, we find it useful to break apart the largest input ranges into multiple ranges of comparable width to the average input range. `subdivideGRanges` requires an input *GRanges* object and returns a *GRanges* object.

```
> library(exomeCopy)
> gr <- GRanges(seqname="seq1",IRanges(start=1,end=345))
> subdivideGRanges(gr)

GRanges with 3 ranges and 0 elementMetadata values:
      seqnames     ranges strand
         <Rle>  <IRanges>  <Rle>
  [1]     seq1 [  1, 115]      *
  [2]     seq1 [116, 230]      *
  [3]     seq1 [231, 345]      *
  ---
  seqlengths:
```

```
seq1
  NA
```

The default setting of `subdivideGRanges` is to divide an input range into ranges around $s = 100$bp, which is slightly less than the average exon width. Specifically, an input range of width w will be divided evenly into $\max(1, \lfloor w/s \rfloor)$ regions. We can visualize the effect of `subdivideGRanges` on ranges of increasing width.

```
>   plot(0,0,xlim=c(0,500),ylim=c(0,25),type="n",yaxt="n",ylab="",
+         xlab="width of input GRanges object",
+         main="Effect of subdivideGRanges")
>   abline(v=1:5*100,col="grey")
>   for (i in 1:24) {
+     gr <- GRanges(seqname="chr1",IRanges(start=1,width=(i*20)))
+     sbd.gr <- subdivideGRanges(gr)
+     arrows(start(sbd.gr),rep(i,length(sbd.gr)),end(sbd.gr),
+            rep(i,length(sbd.gr)),length=.04,angle=90,code=3)
+   }
```



Here we demonstrate reading in a targeted region BED file, converting to a *GRanges* object and the result from calling `subdivideGRanges`. Note that if the targeted region is read in from a BED file, one should add 1 to the starting position for representation as a *GRanges* object.

```
> target.file <- system.file("extdata","targets.bed",package="exomeCopy")
> target.df <- read.delim(target.file,header=FALSE,
+                         col.names=c("seqname","start","end"))
> target <- GRanges(seqname=target.df$seqname,
+                   IRanges(start=target.df$start+1,end=target.df$end))
> target
```

```
GRanges with 5 ranges and 0 elementMetadata values:
      seqnames     ranges strand
         <Rle>  <IRanges>  <Rle>
  [1]     seq1 [101, 250]      *
  [2]     seq1 [301, 650]      *
  [3]     seq2 [  1, 150]      *
  [4]     seq2 [401, 550]      *
  [5]     seq2 [701, 750]      *
  ---
  seqlengths:
   seq1 seq2
     NA   NA
```

```
> target.sub <- subdivideGRanges(target)
> target.sub
```

```
GRanges with 7 ranges and 0 elementMetadata values:
      seqnames     ranges strand
         <Rle>  <IRanges>  <Rle>
  [1]     seq1 [101, 250]      *
  [2]     seq1 [301, 417]      *
  [3]     seq1 [418, 533]      *
  [4]     seq1 [534, 650]      *
  [5]     seq2 [  1, 150]      *
  [6]     seq2 [401, 550]      *
  [7]     seq2 [701, 750]      *
  ---
  seqlengths:
   seq1 seq2
     NA   NA
```

2.2 Counting reads in genomic ranges

countBamInGRanges allows the user to count reads from a BAM read mapping file in genomic ranges covering the targeted region. The function takes as input the BAM filename and a *GRanges* object. It returns a vector of counts, representing the number of sequenced read starts (leftmost position regardless of strand) with mapping quality above a minimum threshold (default of 1) for each genomic range. Users should make sure the sequence names in the *GRanges* object are the same as the sequence names in the BAM file (which can be listed using scanBamHeader in the Rsamtools package). The BAM file requires a associated index file (see the man page for indexBam in the Rsamtools package). We will count reads using the subdivided genomic ranges in target.sub and store the counts as a new value column, sample1.

```
> bam.file <- system.file("extdata","mapping.bam",package="exomeCopy")
> scanBamHeader(bam.file)[[1]]$targets
```

```
seq1 seq2
 800  800
```

```
> levels(seqnames(target.sub))
```

```
[1] "seq1" "seq2"
```

```
> rdata <- RangedData(space=seqnames(target.sub),ranges=ranges(target.sub))
> rdata[["sample1"]] <- countBamInGRanges(bam.file,target.sub)
> rdata
```

```
RangedData with 7 rows and 1 value column across 2 spaces
     space     ranges |   sample1
  <factor>  <IRanges> | <numeric>
1     seq1 [101, 250] |        73
2     seq1 [301, 417] |        59
3     seq1 [418, 533] |        61
4     seq1 [534, 650] |        54
5     seq2 [  1, 150] |        80
6     seq2 [401, 550] |        69
7     seq2 [701, 750] |        31
```

2.3 Calculating GC-content

exomeCopy can model read counts from samples which are not perfectly correlated with background read depth using GC-content (ratio of G and C bases to total number of bases). The GC-content of DNA fragments is known to be a factor in the efficiency of high-throughput sequencing. Using `scanFa` in the Rsamtools package and a FASTA file of the reference genome, we can obtain a *DNAStringSet* object for the DNA sequence of the genomic ranges. Then `letterFrequency` in the Biostrings package can be used to tally the GC-content.

```
> reference.file <- system.file("extdata","reference.fa",package="exomeCopy")
> target.dnastringset <- scanFa(reference.file,target.sub)
> getGCcontent <- function(x) {
+   GC.count <- letterFrequency(x,"GC")
+   all.count <- letterFrequency(x,"ATGC")
+   as.vector(ifelse(all.count==0,NA,GC.count/all.count))
+ }
> rdata[["GC"]] <- getGCcontent(target.dnastringset)
> rdata
```

```
RangedData with 7 rows and 2 value columns across 2 spaces
     space     ranges |   sample1        GC
  <factor>  <IRanges> | <numeric> <numeric>
1     seq1 [101, 250] |        73 0.4533333
2     seq1 [301, 417] |        59 0.4273504
3     seq1 [418, 533] |        61 0.5689655
4     seq1 [534, 650] |        54 0.5213675
5     seq2 [  1, 150] |        80 0.5733333
6     seq2 [401, 550] |        69 0.4800000
7     seq2 [701, 750] |        31 0.3600000
```

In the following sections, we will continue with a dataset constructed using real targeted regions and exome sequencing data.

3 Exome sequencing data from 1000 Genomes Project

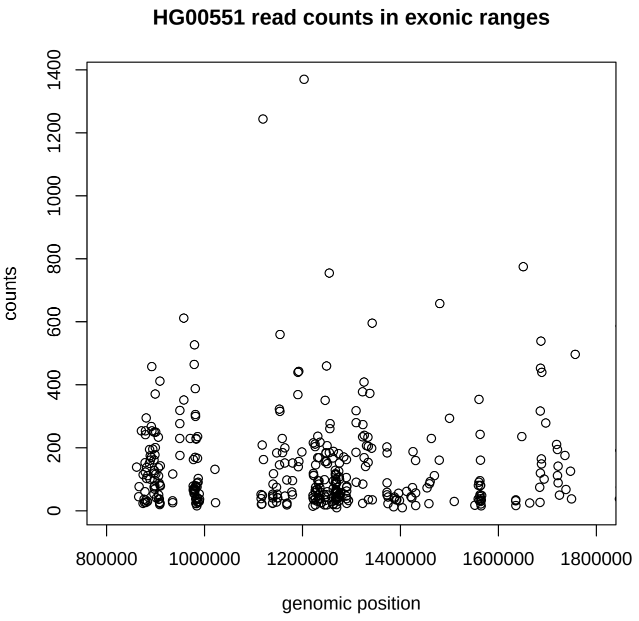
For demonstrating the use of exomeCopy , we have provided a sample dataset, exomecounts, of exome sequencing read counts. The genomic ranges used in this dataset are generated from a small subset of the CCDS regions on chromosome 1 [Pruitt et al., 2009]. The regions are downloaded from the hg19 tables of the UCSC Genome Browser (http://genome.ucsc.edu/cgi-bin/hgGateway). Alternatively, one could use the GenomicFeatures package to download the CCDS regions. The original CCDS regions are subdivided using subdivideGRanges with default settings as in the example code above. The CCDS regions are convenient to use as genomic ranges for CNV detection in exome sequencing data, as they are often in the center of the targeted region in exome enrichment protocols and tend to have less variable coverage than the flanking regions.

The read counts are taken from exome enriched, paired-end sequencing data of the 1000 Genomes Project for 16 samples of the PUR population [1000 Genomes Project Consortium, 2010]. The BAM read mapping files and descriptions of the experiments are available at the 1000 Genomes Project website (http://www.1000genomes.org/data). The ftp addresses used are listed in the file 1000Genomes_files.txt in the extdata directory. The sample names are included as column names in the provided datatset.

```
> data(exomecounts)
> dim(exomecounts)

[1] 1000   17

> exomecounts[1:5,1:3]

RangedData with 5 rows and 3 value columns across 1 space
     space           ranges |        GC   HG00551   HG00641
  <factor>        <IRanges> | <numeric> <numeric> <numeric>
1     chr1 [861322, 861393] |    0.6389       139       223
2     chr1 [865535, 865716] |    0.6484        45        90
3     chr1 [866419, 866469] |    0.5882        77       123
4     chr1 [871152, 871276] |    0.6480       254       285
5     chr1 [874420, 874509] |    0.6111        24        40
```

The genomic ranges in exomecounts have been filtered such that only ranges with nonzero read count over the 16 samples are retained. The range of the 1000 genomic ranges is from 0.8 to 7.8 Mb on chromosome 1. Plotting the counts for one sample in a region of 1 Mb, one can observe both the irregular spacing of the ranges as well as the non-uniformities in read counts per range.

```
> plot(start(exomecounts),exomecounts$HG00551,xlim=c(0.8e6,1.8e6),
+      xlab="genomic position",ylab="counts",
+      main="HG00551 read counts in exonic ranges")
```



3.1 Generating background read depth

In order to run exomeCopy , we first generate background read depth. We extract a read counts data frame from the *RangedData* object and divide each sample by its mean read count (column mean). Our read depth background is the median of these normalized read counts over the 16 samples (row median). We could at this point also store the standard deviation of the normalized read counts for a measure of read depth deviation at each range, but we will not use this information in this vignette.

```
> exome.samples <- grep("HG.+",colnames(exomecounts),value=TRUE)
> sample.columns <- colnames(exomecounts) %in% exome.samples
> C <- as.data.frame(unlist(values(exomecounts)[,sample.columns]))
> C.norm <- sweep(C,2,colMeans(C),"/")
> exomecounts[["bg"]] <- apply(C.norm,1,median)
```

The relationship between read counts and GC-content over the ranges varies across protocols and samples. It can be roughly approximated per sample using second-order polynomial terms of GC-content. We store the square of GC-content as a new value column. Other functions of GC-content could be used as well. We also store the width of the ranges as a value column.

```
> exomecounts[["GC.sq"]] <- exomecounts$GC^2
> exomecounts[["width"]] <- width(exomecounts)
```

4 Brief introduction of the model

exomeCopy models the sample read counts on one chromosome, O, as emitted observations of a hidden Markov model (HMM), where the hidden state is the copy number of the sample. The emission distributions f are modeled with negative binomial distributions, as the read counts from high-throughput sequencing are often overdispersed for the Poisson distribution.

$$f \sim \mathrm{NB}(O_t, \mu_{ti}, \phi)$$

$$\mu_{ti} = \frac{S_i}{d}(x_{t*}\beta)$$

The mean parameter, μ_{ti}, for genomic range t and hidden state i is a product of the possible copy number state S_i over the expected copy number d and an estimate of the positional effects on read depth $(x_{t*}\beta)$, where x_{t*} is the t-th row of X and β is a column vector of coefficients. The estimated positional effect is a linear combination of background read depth, GC-content, range width, and any other useful covariates which are stored in the matrix X, with a row for each range and a column for each covariate. The coefficients β are fit by the model, using the forward equations to assess the likelihood of the HMM over all hidden state paths. In this way, the normalization and segmentation steps are combined into one step of maximizing the likelihood of the parameters given the data. The Viterbi algorithm is then applied to provide the most likely path.

Table 1: Summary of notation

O_t	observed count of reads in the t-th genomic range
f	the emission distribution for read counts
μ_{ti}	the mean parameter for f at range t in copy state i
ϕ	the dispersion parameter for f
S_i	the copy number for state i ($S_i \in \{0, 1, 2, \ldots\}$)
d	the expected background copy number (2 for diploid, 1 for haploid)
X	the matrix of covariates for estimating μ
Y	the matrix of covariates for estimating ϕ
β	the fitted coefficients for estimating μ
γ	the fitted coefficients for estimating ϕ

The base model uses a scalar estimate for the dispersion parameter ϕ (equivalent to `1/size` in the function `dnbinom`). An extension of this model also tries to fit the variance using positional information such as the standard deviation of the background read depth stored in a matrix Y.

$$f \sim \mathrm{NB}(O_t, \mu_{ti}, \phi_t)$$

$$\phi_t = y_{t*}\gamma$$

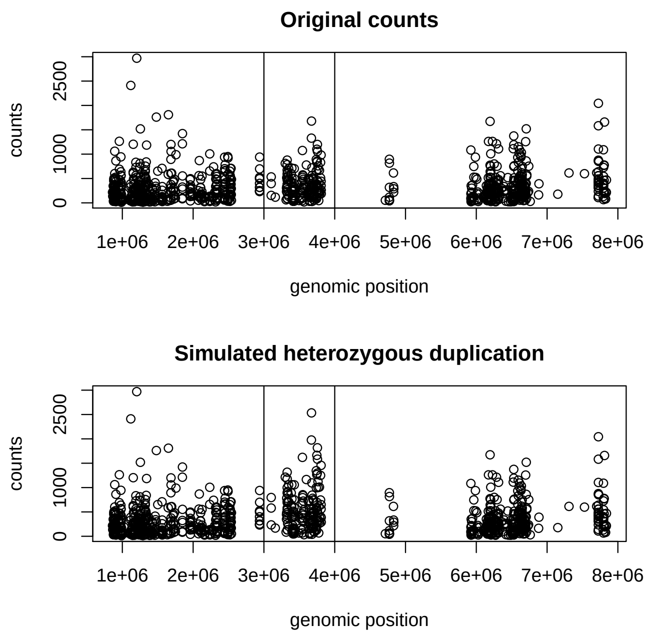
Both μ_{ti} and ϕ_t must be positive, so negative estimates are replaced with a small positive number (`1e-8`).

5 Running exomeCopy on simulated CNVs

Next we simulate CNVs in 4 samples representing regions with a copy number of 0,1,3 and 4, relative to a background copy number of 2. This is accomplished by selecting a fraction of the reads contained in the CNV bounds and removing or doubling them. The new counts with simulated CNVs are added as new value columns to the *RangedData* object. We then plot the original counts and the counts within a simulated heterozygous duplication.

```
> set.seed(2)
> cnv.type <- c("hom.del","het.del","het.dup","hom.dup")
> cnv.probs <- c(.99,.5,.5,.95)
> cnv.mult <- c(-1,-1,1,1)
> bounds <- IRanges(start=3e6,end=4e6)
> for (i in 1:4) {
+   samplename <- exome.samples[i]
+   contained <- unlist(ranges(exomecounts)) %in% bounds
+   O <- exomecounts[[samplename]]
+   O[contained] <- O[contained] + (cnv.mult[i] *
+     rbinom(sum(contained),prob=cnv.probs[i],size=O[contained]))
+   exomecounts[[paste(samplename,cnv.type[i],sep=".")]] <- O
+ }
```

```
> par(mfrow=c(2,1),mar=c(5,4,3,2))
> plot(start(exomecounts),exomecounts[["HG00731"]],
+      xlab="genomic position",ylab="counts",main="Original counts")
> abline(v=c(start(bounds),end(bounds)))
> plot(start(exomecounts),exomecounts[["HG00731.het.dup"]],
+      xlab="genomic position",ylab="counts",
+      main="Simulated heterozygous duplication")
> abline(v=c(start(bounds),end(bounds)))
```



5.1 Running exomeCopy and inspecting the segmentation

We can now run exomeCopy using the read counts for one of the simulated CNV samples. Later we will show how to write a simple wrapper function to loop the `exomeCopy` function over multiple chromosomes and samples. We specify the possible copy number values with `S` and the expected copy number state with `d`.

```
> fit <- exomeCopy(exomecounts["chr1"],sample.name="HG00731.het.dup",
+                  X.names=c("bg","GC","GC.sq","width"),S=0:6,d=2)
> show(fit)
```

```
ExomeCopy object
type: exomeCopy
percent normal state: 83.5%
```

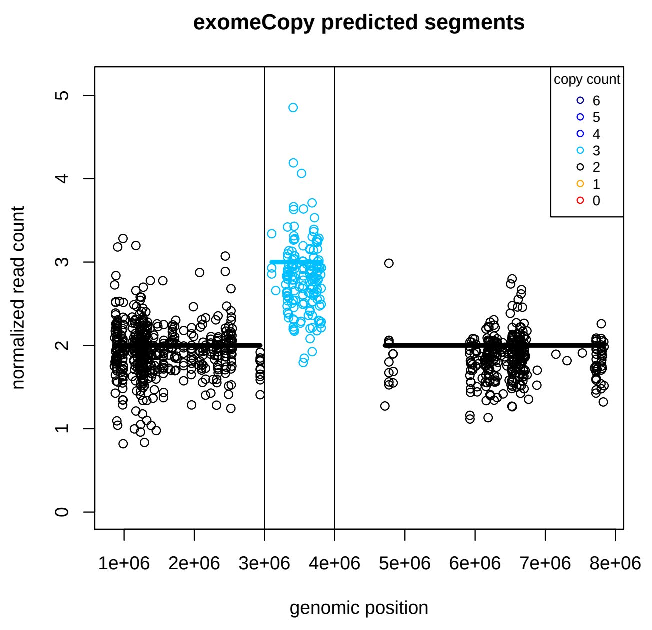
After fitting, we call the function `copyCountSegments` on the *ExomeCopy* object, which provides the segmentation with the predicted copy number and the number of input genomic ranges contained within each segment. exomeCopy correctly identifies the segment between 3-4 Mb as having copy number 3 against the background copy number 2.

```
> copyCountSegments(fit)
```

```
RangedData with 3 rows and 2 value columns across 1 space
     space              ranges | copy.count   nranges
  <factor>           <IRanges> |  <integer> <numeric>
1     chr1 [ 861322, 2939384] |          2       503
2     chr1 [3102689, 3809560] |          3       165
3     chr1 [4715486, 7838229] |          2       332
```

Calling `plot` on the *ExomeCopy* object draws segments of constant predicted copy number as colored horizontal lines and normalized read counts as points. The vertical lines indicate the start and end of the simulated CNV of copy number 3.

```
> cols <- c("red","orange","black","deepskyblue","blue","blue2","blue4")
> plot(fit,col=cols)
> abline(v=c(start(bounds),end(bounds)))
```



5.2 Looping exomeCopy over multiple chromosomes/samples

In order to apply exomeCopy to a full dataset of multiple chromosomes and samples, we write a wrapper function runExomeCopy. This allows for distribution of jobs across workstations, for example.

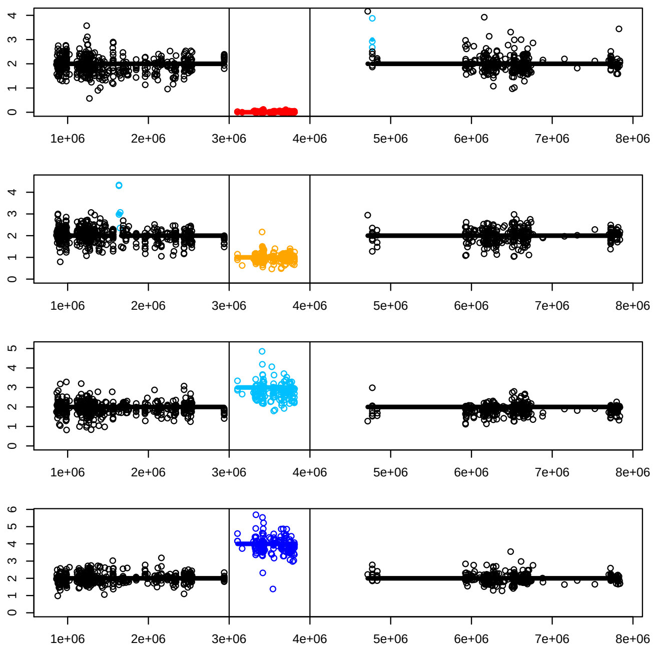
```
> runExomeCopy <- function(idx,rdata,seq.loop,sample.loop) {
+   seq.name <- seq.loop[idx]
+   sample.name <- sample.loop[idx]
+   exomeCopy(rdata[seq.name],sample.name,
+             X.names=c("bg","GC","GC.sq","width"),S=0:6,d=2)
+ }
```

We can now run exomeCopy using either lapply or using a function like clusterApplyLB from the snow package. We define an order of chromosome-sample pairs by constructing variables seq.loop and sample.loop with the rep function. By progressing through these two vectors, we predict CNVs in all chromosomes for all samples. We apply runExomeCopy over the 4 samples with simulated CNVs and across 1 chromosome, though this code could be used across multiple chromosomes as well.

```
> seqs <- c("chr1")
> samples <- paste(exome.samples[1:4],cnv.type,sep=".")
> nseqs <- length(seqs)
> nsamples <- length(samples)
> seq.loop <- rep(seqs,times=nsamples)
> sample.loop <- rep(samples,each=nseqs)
> fit.list <- lapply(seq_len(nseqs*nsamples),
+                    runExomeCopy,exomecounts,seq.loop,sample.loop)
```

We can visualize the segments of constant predicted copy number for all simulated CNV samples. The vertical lines indicate the start and end of the simulated CNVs.

```
> par(mfrow=c(4,1),mar=c(3,3,1,1))
> for (i in 1:4) {
+   plot(fit.list[[i]],main="",xlab="",ylab="",show.legend=FALSE,col=cols)
+   abline(v=c(start(bounds),end(bounds)))
+ }
```



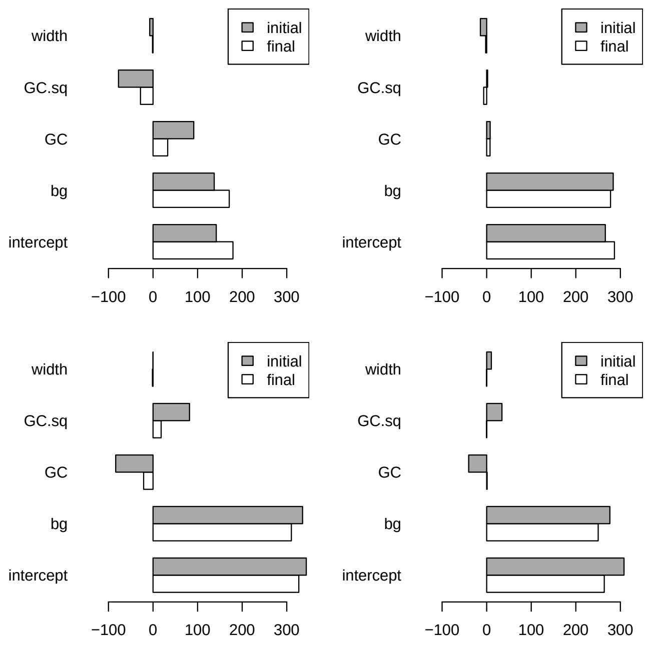
5.3 Inspecting model parameters

Finally, we can inspect a number of fitted parameters in the *ExomeCopy* object, such as the β vector which is adjusted in optimizing the likelihood of the HMM. The β vector is initialized using a linear regression of counts on the covariates, and this and other initial parameter values can be accessed in the `init.par` slot of the *ExomeCopy* object. These coefficients are adjusted while optimizing the likelihood of the HMM. The final β vector and other final parameter values are accessible in the `final.par` slot.

```
> fit.list[[1]]@init.par$beta.hat

 intercept         bg         GC      GC.sq      width
141.820000 137.231101  91.349321 -77.486210  -7.434786

> fit.list[[1]]@final.par$beta

 intercept         bg         GC      GC.sq      width
179.323912 170.942296  32.850119 -28.145603  -1.168821
```

If we plot the initial and final β vectors from the 4 samples with simulated CNVs, we can observe that the β vector varies across patients and that the coefficients which maximize the full HMM are not the same as the initial estimates from regression.

```
> par(mfrow=c(2,2),mar=c(3,5,1,1))
> for (i in 1:4) {
+   barplot(rbind(fit.list[[i]]@final.par$beta,fit.list[[i]]@init.par$beta.hat),
+           horiz=TRUE,las=1,beside=TRUE,col=c("white","darkgrey"),
+           xlim=c(-150,350))
+   legend("topright",legend=c("initial","final"),fill=c("darkgrey","white"))
+ }
```



6 Session info

```
> sessionInfo()

R version 2.14.2 (2012-02-29)
Platform: x86_64-unknown-linux-gnu (64-bit)

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

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

other attached packages:
[1] exomeCopy_1.0.3     Rsamtools_1.6.3     Biostrings_2.22.0
[4] GenomicRanges_1.6.7 IRanges_1.12.6

loaded via a namespace (and not attached):
[1] BSgenome_1.22.0    RCurl_1.91-1       XML_3.9-4          bitops_1.0-4.1
[5] rtracklayer_1.14.4 tools_2.14.2       zlibbioc_1.0.1
```

References

1000 Genomes Project Consortium. A map of human genome variation from population-scale sequencing. *Nature*, 467(7319):1061–1073, October 2010. ISSN 1476-4687. doi: 10.1038/nature09534. URL `http://dx.doi.org/10.1038/nature09534`.

Michael I. Love, Alena Myšičková, Ruping Sun, Vera Kalscheuer, Martin Vingron, and Stefan A. Haas. Modeling Read Counts for CNV Detection in Exome Sequencing Data. *Statistical Applications in Genetics and Molecular Biology*, 10(1), November 2011. ISSN 1544-6115. doi: 10.2202/1544-6115.1732. URL `http://cmb.molgen.mpg.de/publications/Love_2011_exomeCopy.pdf`.

Kim D. Pruitt, Jennifer Harrow, Rachel A. Harte, Craig Wallin, Mark Diekhans, Donna R. Maglott, Steve Searle, Catherine M. Farrell, Jane E. Loveland, Barbara J. Ruef, Elizabeth Hart, Marie-Marthe M. Suner, Melissa J. Landrum, Bronwen Aken, Sarah Ayling, Robert Baertsch, Julio Fernandez-Banet, Joshua L. Cherry, Val Curwen, Michael Dicuccio, Manolis Kellis, Jennifer Lee, Michael F. Lin, Michael Schuster, Andrew Shkeda, Clara Amid, Garth Brown, Oksana Dukhanina, Adam Frankish, Jennifer Hart, Bonnie L. Maidak, Jonathan Mudge, Michael R. Murphy, Terence Murphy, Jeena Rajan, Bhanu Rajput, Lillian D. Riddick, Catherine Snow, Charles Steward, David Webb, Janet A. Weber, Laurens Wilming, Wenyu Wu, Ewan Birney, David Haussler, Tim Hubbard, James Ostell, Richard Durbin, and David Lipman. The consensus coding sequence (CCDS) project: Identifying a common protein-coding gene set for the human and mouse genomes. *Genome research*, 19(7):1316–1323, July 2009. ISSN 1088-9051. doi: 10.1101/gr.080531.108. URL `http://dx.doi.org/10.1101/gr.080531.108`.