Wave correction for arrays

Eitan Halper-Stromberg and Robert B. Scharpf

June 29, 2013

Abstract

`ArrayTV` is a GC correction algorithm for microarray data designed to mitigate the appearance of waves. This work is inspired by a method for mitigating waves in sequencing data [1]. We find the genomic window for computing GC that best captures the dependence of signal intensity on GC content for each array using the total variance distance statistic (TV score). The correction each probe receives is the mean signal intensity of all probes sharing the same local GC. We center windows at the starting position of each probe.

1 Adjusting copy number estimates for GC waves

Importing data. The wave correction methods in `ArrayTV` handle simple matrices with rows indexing markers and columns indexing samples, as well as more complex data structures commonly used to process Affymetrix and Illumina platforms made available in the `oligoClasses` package. In this vignette, we illustrate our approach with a single female lymphoblastoid cell line sample assayed on the Agilent 1M , Affymetrix 2.7M, and NimbleGen 2.1M platforms (available in full from http://rafalab.jhsph.edu/cnvcomp, spike-in tube 2)[2]. For each platform, we provide a matrix. The rows of the matrix correspond to markers on chromosome 15 and the columns correspond to the physical position of the first nucleotide of the marker using UCSC build hg18 (note: for genotyping arrays, the position should be adjusted to the start of the marker and not the location of the SNP) and preprocessed signal intensities that we expect to be proportional to the copy number as well as sequence artifacts such as GC content. Agilent and NimbleGen signal intensities were preprocessed by taking the log ratio of loess normalized raw signal. Affymetrix performed their own preprocessing to obtain log R ratios. Hereafter, we refer to the preprocessed signal as M values. To keep the size of this package small, we provide the M values as integers ($M \times 1000$). In the following code, we load the data for each platform, inspect the first few rows, and transform the M values for each platform to the original scale.

```
> library(ArrayTV)
> path <- system.file("extdata", package="ArrayTV")
> load(file.path(path, "array_logratios.rda"))
> head(agilent)

     position    M
[1,] 18362555  233
[2,] 18427151  -29
[3,] 18432556   50
[4,] 18450562    7
[5,] 18452491   14
[6,] 18692865 -604

> head(affymetrix)

     position    M
[1,]      601  -47
[2,]     1189 -131
```

```
[3,]     3590 -290
[4,]     4668 -376
[5,]     5942  109
[6,]     6677 -405
```

```
> head(nimblegen)
```

```
     position    M
[1,] 18305309   55
[2,] 18308562 -173
[3,] 18314533 -115
[4,] 18317177  148
[5,] 18318410   -4
[6,] 18321680   -6
```

```
> agilent[, "M"] <- agilent[, "M"]/1000
> affymetrix[, "M"] <- affymetrix[, "M"]/1000
> nimblegen[, "M"] <- nimblegen[, "M"]/1000
```

We will generate total variance scores (TV scores) for 30 window sizes to determine the optimal window(s) for GC correction. Again, we assume that the positions denote the start location of each probe. While our analysis is limited to chromosome 15 for practical constraints, in practice we recommend implementing the wave correction on all autosomes simultaneously. Next, we specify `increms` and `wins` to indicate the size of the genomic intervals surrounding the markers from which GC content will be computed. Here each of the 3 entries in the vector `increms` specifies an increment size, with the corresponding entries in `wins` specifyng the maximum window size to evaluate for that increment. In this case evaluated windows will be of sizes: 10,20,30,..100,1e3,2e3,3e3,...10e3,1e5,2e5,3e5,...1e6

```
> increms <- c(10,1000,100e3)
> wins <- c(100,10e3,1e6)
```

To enable parallelization, execute the following unevaluated code chunk. Parallelization is by chromosome for calculation of GC content in each window and by array for calculation of correction values.

```
> library(doMC)
> registerDoMC(nodes)
```

Evaluating TV scores as a function of window size for computing GC. The class of the first argument to the `gcCorrect` method determines the dispatch to the low-level function `gcCorrectMain`. Any of the arguments to `gcCorrectMain` can be passed through the `...` operator and we refer the user to the documentation for `gcCorrectMain` for additional arguments and `gcCorrect` for the classes of objects handled by the `gcCorrect` method. Computing the GC content of the sequence for each marker requires that the appropriate BSgenome package is loaded. For example, here the package `BSgenome.Hsapiens.UCSC.hg18` is loaded internally. The following (unevaluated) codechunk performs GC correction on the M values for the Nimblegen, Agilent, and Affymetrix platforms, respectively.

```
> nimcM1List <- gcCorrect(object=nimblegen[, "M", drop=FALSE],
+                         chr=rep("chr15", nrow(nimblegen)),
+                         starts=nimblegen[, "position"],
+                         increms=increms,
+                         maxwins=wins,
+                         build='hg18')
> agcM1List <- gcCorrect(agilent[, "M", drop=FALSE],
+                        chr=rep("chr15", nrow(agilent)),
+                        starts=agilent[, "position"],
+                        increms=increms,
```

```
+                         maxwins=wins,
+                         build="hg18")
> afcM1List <- gcCorrect(affymetrix[, "M", drop=FALSE],
+                         chr=rep("chr15", nrow(affymetrix)),
+                         starts=affymetrix[,"position"],
+                         increms=increms,
+                         maxwins=wins,
+                         build="hg18")
```

We load the previously computed results, including TV scores, from file

```
> load(file.path(system.file("extdata", package="ArrayTV"), "tvScoresAndCorrected.rda"))
```

In the following code chunk, we plot the TV scores (Figure 1). We concatenate the results from each platform into a list and pass this list into the plotTV function. This function uses the increms and wins vectors to organize plotting based on the size of the genomic windows used to compute GC (small, medium, and large)

```
> plotTV<- function(tvScoreList,increms,maxwins,windowsize,cols,...){
+   for(ii in seq_along(increms)){
+     if(!is.null(tvScoreList[['tvScore']])) tvScoreList <- list(tvScoreList)
+     for(jj in seq_along(tvScoreList)){
+       toplot<-tvScoreList[[jj]][['tvScore']][match(seq(increms[ii],maxwins[ii],increms[ii]),
+         as.numeric(rownames(tvScoreList[[jj]][['tvScore']]))),,drop=FALSE]
+       if(jj == 1){
+         matplot(rownames(toplot),toplot,
+         col=cols[jj],...)
+       }else{
+         matpoints(rownames(toplot),toplot,col=cols[jj],...)
+       }
+
+       if(ii==1){
+         axis(2, at=pretty(c(0,0.1), 6), outer=TRUE)
+         legend('top',legend=c('NimbleGen','Agilent','Affy'),
+         col=c('blue','green','red'),pch=20)
+       }
+     }
+
+   }
+ }
> par(mfrow=c(1,3), mar=c(2.5,0.05,1.6,0.05), mgp=c(1.5,.5,0),
+ oma=c(0,4,4,0), las=1)
> windowsize=c('small','medium','large')
> plotTV(list(nimcM1List,agcM1List,afcM1List),increms,wins,windowsize,
+ cols=c('blue','green','red'),pch=20,cex=2,ylab='tv score',xlab="window extension from center",
+ yaxt="n",type="b",ylim=c(0,.1))
```

View corrected and uncorrected M values

Our GC-adjusted signal was computed when we called gcCorrect and is now stored in nimcM1List,agcM1List, and afcM1List. Here we create a dataframe to store corrected and uncorrected signal, ahead of plotting.

```
> wave.df <- data.frame(position=c(rep(nimblegen[, "position"]/1e6, 2),
+                         rep(agilent[, "position"]/1e6, 2),
+                         rep(affymetrix[, "position"]/1e6,2)),
```

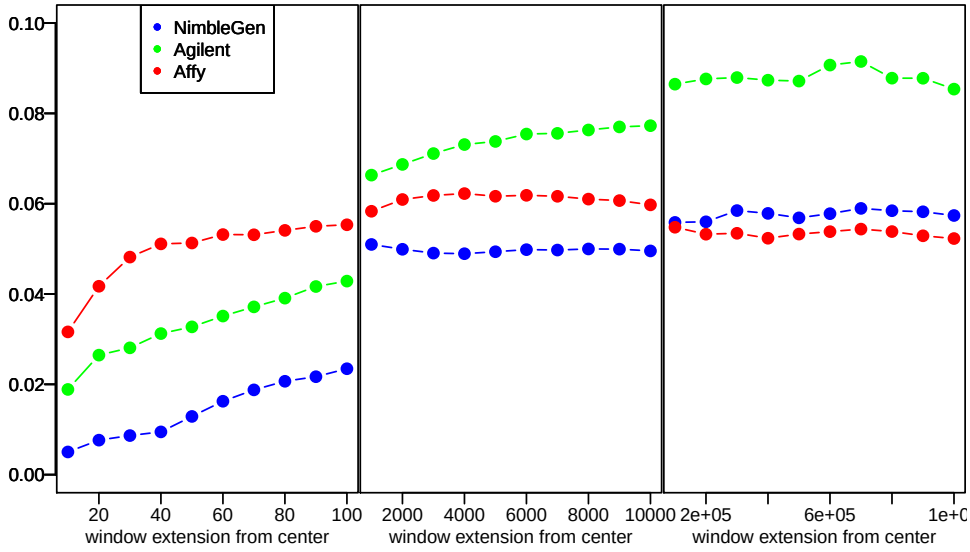

Figure 1: TV scores for three array platforms (colors) in three sets of windows (columns), ranged 10..100, 1,000-10,000 and 100,000-1,000,000

```
+                       r=c(nimblegen[, "M"],nimcM1List[['correctedVals']],
+                       agilent[,"M"],agcM1List[['correctedVals']],
+                       affymetrix[,"M"], afcM1List[['correctedVals']]),
+                       platform=rep(c(rep("Nimblegen", 2),
+                       rep("Agilent", 2),
+                       rep("Affymetrix", 2)), c(rep(length(nimblegen[, "M"]),2),
+                                                rep(length(agilent[,"M"]),2),
+                                                rep(length(affymetrix[,"M"]),2))),
+                       wave.corr=c(rep("raw", length(nimblegen[, "M"])),
+                       rep("ArrayTV", length(nimblegen[, "M"])),
+                       rep("raw", length(agilent[,"M"])),
+                       rep("ArrayTV", length(agilent[,"M"])),
+                       rep("raw", length(affymetrix[,"M"])),
+                       rep("ArrayTV",length(affymetrix[,"M"]))))
> wave.df$platform <- factor(wave.df$platform, levels=c("Nimblegen",
+                                                       "Agilent", "Affymetrix"))
> wave.df$wave.corr <- factor(wave.df$wave.corr, levels=c("raw", "ArrayTV"))
```

We use a `lattice` display to visualize the M values before and after correction for the three platforms (Figure 2)

```
>     ## For each of the 3 example platforms, the uncorrected signal appears immediately above the
>     ## corrected signal, across chromosome 15. A smoothed loess line is drawn through each.
> require("lattice") && require("latticeExtra")

[1] TRUE

> p <- xyplot(r~position|platform+wave.corr, wave.df, pch=".", col="gray",
+             ylim=c(-2,1.5), xlim=c(18,102),ylab="M", xlab="position (Mb) on chr15",
```

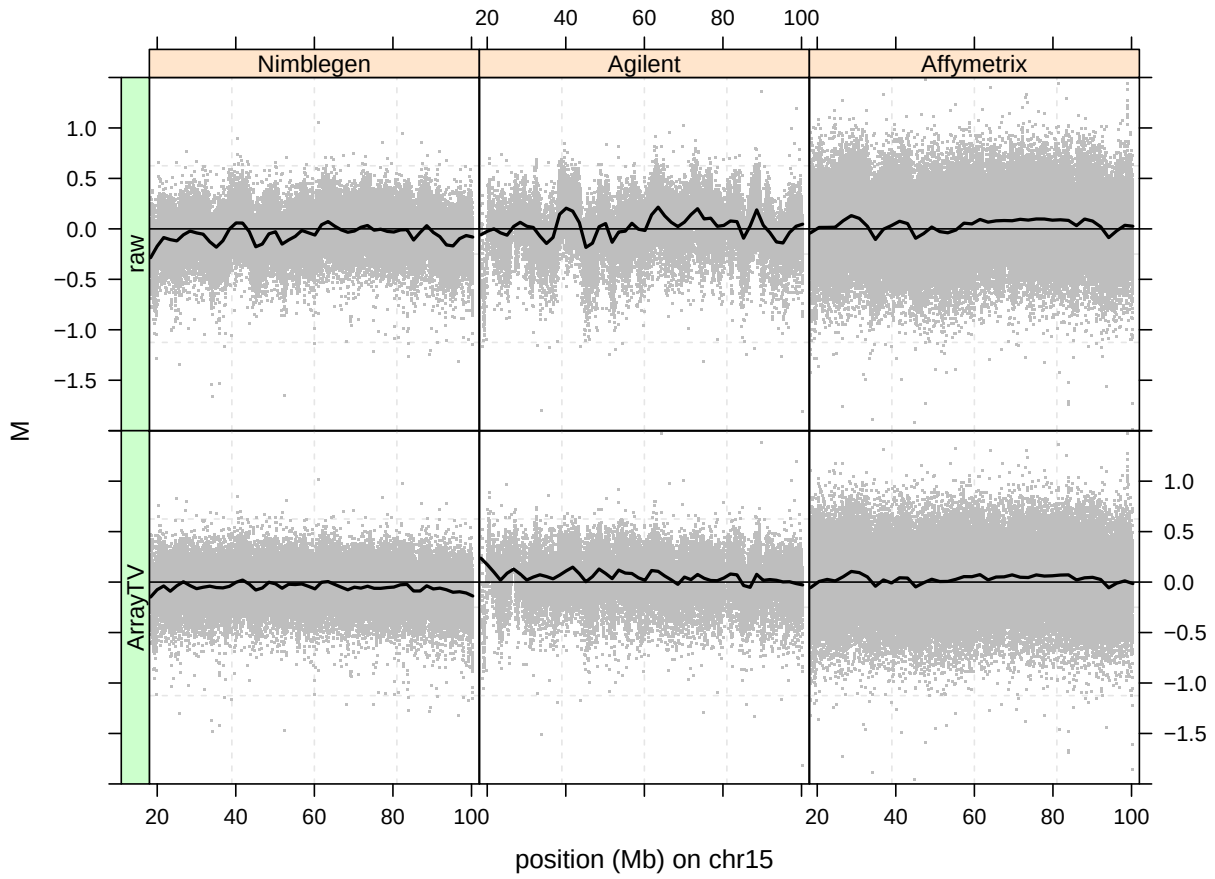

Figure 2: Comparison of M values before and after `ArrayTV` wave correction (rows) for three array platforms (columns).

```
+             scales=list(y=list(tick.number=8)),
+             panel=function(x,y,subscripts,...){
+                     panel.grid(lty=2)
+                     panel.xyplot(x, y, ...)
+                     panel.abline(h=0)
+                     panel.loess(x, y, span=1/20, lwd=2, col="black")
+             }, par.strip.text=list(cex=0.9),
+             layout=c(2,3),
+             as.table=TRUE)

> p2 <- useOuterStrips(p)
> print(p2)
```

2 View a SpikeIn

In this section, we use CBS to segment corrected and uncorrected signal within a region containing a known copy number amplification. The true M value for the spike-in is 2.5, and our goal is to use CBS to recover the spike-in signal that is partly obscured by waves in the uncorrected Ms. The amplification was spiked in

to each of the three array platforms discussed previously. In the following codechunk, we collect the raw and GC-corrected M values in a list object, and perform CBS segmentation on the elements of the list:

```
> if(require("DNAcopy")){
+         cbs.out <- data.frame()
+         cbs.segs <- list()
+         platforms <- c("Nimblegen", "Agilent", "Affymetrix")
+         correction <- c("raw", "ArrayTV")
+         Ms <- list(nimblegen[, "M"], nimcM1List[['correctedVals']],
+                    agilent[, "M"], agcM1List[['correctedVals']],
+                    affymetrix[, "M"], afcM1List[['correctedVals']])
+         starts <- list(nimblegen[, "position"],
+                        agilent[, "position"],
+                        affymetrix[, "position"])
+         k <- 0
+         for(i in 1:3){
+                 for(j in 1:2){
+                         k <- k+1
+                         MsUse <- Ms[[k]]
+                         chrUse <- rep("chr15", length(MsUse))
+                         startsUse <- starts[[i]]
+                         toUse <- !(duplicated(startsUse))
+                         cna1 <- CNA(as.matrix(MsUse[toUse]),
+                                     chrUse[toUse],
+                                     startsUse[toUse],
+                                     data.type="logratio",
+                                     paste(platforms[i], correction[j], sep="_"))
+                         cbs.segs[[k]] <- segment(cna1, verbose=1,min.width = 5)
+                 }
+             }
+         cbs.out <- do.call("rbind", lapply(cbs.segs, "[[", "output"))
+     }
```

As CBS segmentation of these genomes is time-consuming, we have saved the results from the above computation that can be loaded as follows:

```
> load(file.path(path, "cbs_out.rda"))
```

To facilitate downstream analyses such as visualization of the results, we coerce the results from the CBS algorithm to an object of class `GRanges`.

```
> ids <- cbs.out$ID
> ## create vectors indicating corrected/uncorrected and platform for each cbs segment
>
> platforms <- gsub("_raw", "", ids)
> platforms <- gsub("_ArrayTV", "", platforms)
> correction <- gsub("Nimblegen_", "", ids)
> correction <- gsub("Affymetrix_", "", correction)
> correction <- gsub("Agilent_", "", correction)
> ## store cbs segments in a GRanges object with designations for corrected/uncorrected
> ## and platform
>
> cbs.segs <- GRanges(rep("chr14", nrow(cbs.out)),
+                     IRanges(cbs.out$loc.start, cbs.out$loc.end),
+                     numMarkers=cbs.out$num.mark,
+                     seg.mean = cbs.out$seg.mean,
```

```
+                     platform = platforms,
+                     correction=correction)
> ## separate corrected and uncorrected cbs segments int separate objects
>
> cbs.segs.raw <- cbs.segs[cbs.segs$correction=="raw", ]
> cbs.segs.arrayTV <- cbs.segs[cbs.segs$correction=="ArrayTV", ]
> ## create GRanges objects with M values, platform, and
> ## corrected/uncorrected designation
>
> marker.data <- GRanges(rep("chr14", nrow(wave.df)),
+                     IRanges(wave.df$position*1e6, width=1),
+                     numMarkers=1,
+                     seg.mean = wave.df$r,
+                     platform=wave.df$platform,
+                     correction=wave.df$wave.corr)
> marker.raw <- marker.data[marker.data$correction=="raw",]
> marker.arrayTV <- marker.data[marker.data$correction=="ArrayTV",]
> ## populate cbs.seg metadata column by joining M values with CBS segments by location
>
> olaps1 <- findOverlaps(marker.raw, cbs.segs.raw, select="first")
> elementMetadata(marker.raw)$cbs.seg <- cbs.segs.raw$seg.mean[olaps1]
> olaps2 <- findOverlaps(marker.arrayTV, cbs.segs.arrayTV, select="first")
> elementMetadata(marker.arrayTV)$cbs.seg <- cbs.segs.arrayTV$seg.mean[olaps2]
> ## populate SpikeIn metadata column if a marker falls within the Spiked-in region
>
> spikeIn <- IRanges(start=45396506,end=45537976)
> spikeinStart <- 45396506; spikeinEnd <- 45537976
> values(marker.raw)$spikeIn <- marker.raw$cbs.seg
> values(marker.arrayTV)$spikeIn <- marker.arrayTV$cbs.seg
> index1 <- which(is.na(findOverlaps(unlist(ranges(marker.raw)),
+                                    spikeIn,select="first")))
> index2 <- which(is.na(findOverlaps(unlist(ranges(marker.arrayTV)),
+                                    spikeIn,select="first")))
> values(marker.raw)$spikeIn[index1] <- NA
> values(marker.arrayTV)$spikeIn[index2] <- NA
> ## put GRanges objects into a dataframe ahead of plotting
>
> rawd <- as.data.frame(marker.raw)
> atvd <- as.data.frame(marker.arrayTV)
> rawd <- rbind(rawd, atvd)
> rawd$platform <- factor(rawd$platform, levels=c("Nimblegen",
+                                        "Agilent", "Affymetrix"))
> rawd$correction <- factor(rawd$correction, levels=c("raw", "ArrayTV"))
```

Black points in Figure 3 represent signal intensities for each probe, and red line segments correspond to the segmentation of the intensities from CBS. The blue line segments demarcate the spike-in. As in the previous figure, raw signal intensities appear immediately above the corrected intensities for each platform

```
> p2 <- xyplot(seg.mean+cbs.seg+spikeIn~start/1e6|platform+correction, rawd,
+              ylim=c(-1.5, 1.5),
+              pch=".",
+              xlim=c(spikeinStart/1e6-2.4,spikeinEnd/1e6+2.4),
+              ylab="M",
+              xlab="position (Mb)",##pch=c(".", NA,NA),
+              col=c('gray40','black','blue'),
```

```
+               distribute.type=TRUE,
+               type=c('p','l','l'),lwd=c(NA,2,3),
+               scales=list(y=list(tick.number=8)),
+               par.strip.text=list(cex=0.9), layout=c(2,3), as.table=TRUE)
> p3 <- useOuterStrips(p2)

> print(p3)
```

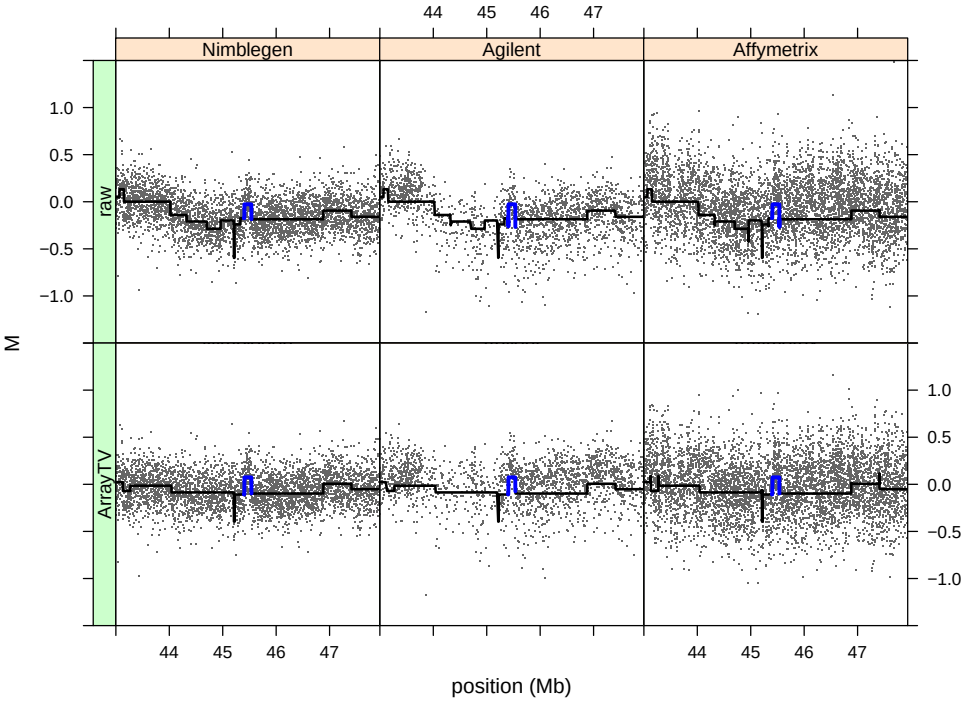

Figure 3: Comparison of CBS segments (lines) and M values (points) before and after `ArrayTV` wave correction (rows) for three array platforms (columns) in the vicinity of a known amplification of copy number 2.5 (blue).

3 Session Information

- R Under development (unstable) (2013-04-15 r62590), `x86_64-apple-darwin12.3.0`
- Locale: `en_US.US-ASCII/en_US.US-ASCII/en_US.US-ASCII/C/en_US.US-ASCII/en_US.US-ASCII`
- Base packages: base, datasets, graphics, grDevices, methods, parallel, stats, utils
- Other packages: ArrayTV 1.1.29, BiocGenerics 0.7.2, BiocInstaller 1.11.3, Biostrings 2.29.9, BSgenome 1.29.0, GenomicRanges 1.13.23, IRanges 1.19.15, lattice 0.20-15, latticeExtra 0.6-24, oligoClasses 1.23.0, RColorBrewer 1.0-5, XVector 0.1.0
- Loaded via a namespace (and not attached): affyio 1.29.0, Biobase 2.21.4, bit 1.1-10, codetools 0.2-8, DNAcopy 1.35.0, ff 2.2-11, foreach 1.4.1, grid 3.1.0, iterators 1.0.6, stats4 3.1.0, tools 3.1.0, zlibbioc 1.7.0

References

[1] Yuval Benjamini and Terence P. Speed. Summarizing and correcting the GC content bias in high-throughput sequencing. *Nucleic Acids Res*, 40(10):e72, May 2012.

[2] Eitan Halper-Stromberg, Laurence Frelin, Ingo Ruczinski, Robert Scharpf, Chunfa Jie, Benilton Carvalho, Haiping Hao, Kurt Hetrick, Anne Jedlicka, Amanda Dziedzic, Kim Doheny, Alan F. Scott, Steve Baylin, Jonathan Pevsner, Forrest Spencer, and Rafael A. Irizarry. Performance assessment of copy number microarray platforms using a spike-in experiment. *Bioinformatics*, 27(8):1052–1060, 2011.