Wave correction for arrays

Eitan Halper-Stromberg and Robert B. Scharpf

May 24, 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 sample assayed on the Agilent, Affymetrix 6.0, and Nimblegen platforms (eitan, is this hapmap? is it the same sample?, do the Nimblegen and Agilent arrays have specific names?). 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 log ratios that we expect to be proportional to the copy number as well as sequence artifacts such as GC content. are these log R ratios or simply log ratios? how were they calculated? Hereafter, we refer to the log ratios 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, expect 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 chromosomes simultaneously (including X and Y, or just the autosomes?) Next, we specify `increms` and `wins` to indicate the size of the genomic intervals surrounding the markers from which GC content will be computed. Please explain what windows will be evaluated based on the selection of increms and wins below.

```
> samplechr <- c('chr15')
> ## Eitan: say what this means
> increms <- c(10,1000,100e3)
> wins <- c(100,10e3,1e6)
```

To enable parallelization, execute the following unevaluated code chunk. is this parallelized by chromosome?

```
> 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. I do not like maxwins and increms. It would be much simpler just to pass the vector of window sizes we want to evaluate? Why not just return a list here: a matrix of tv scores (rows index scores, columns are size of window and TV score), the optimal window size (integer), the higest TV score, the GC values corresponding to the highest tv score, and the adjusted M values corresponding to the highest tv score? get rid of the returnOnly argument. The gcCorrectMain should be able to figure out what we want to do without passing 'samplechr'. The following codechunk is unevaluated.

```
> nimtvScores <- gcCorrect(object=nimblegen[, "M", drop=FALSE],
+                          chr=rep("chr15", nrow(nimblegen)),
+                          starts=nimblegen[, "position"],
+                          samplechr="chr15", ## would be nice not to have to declare this
+                          increms=increms,
+                          maxwins=wins,
+                          returnOnlyTV=TRUE,
+                          build='hg18')
```

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

We load the previously computed TV scores from file, and create a list of the TV scores organized by the size of the genomic window (small, medium, or large) for computing GC.

```
> load(file.path(system.file("extdata", package="ArrayTV"), "tvscores.rda"))
> tvScoresSplit <- split(data.frame(nimtvScores,agtvScores,aftvScores),rep(1:3,each=10))
```

Define a function plotTV with ... for arguments that can be passed to plot. The visualization should then be accomplished by
par(something)
plotTV(agilentResults)
plotTV(affyResults)
plotTV(nimResults)

In the following code chunk, we plot the TV scores.

```
> 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)
> for(ii in 1:3){
+     plot(rownames(tvScoresSplit[[ii]]),tvScoresSplit[[ii]][,1],col='blue',
+          ylim=c(0,max(unlist(tvScoresSplit))),
+          yaxt="n",
+          xlab='window extension from center',
+          pch=20,
+          cex=2,
+          ylab='tv score',main=windowsize[ii],
+          type="l")
+     if(ii==1){
+             axis(2, at=pretty(c(0,0.1), 6), outer=TRUE)
+             legend('top',legend=c('NimbleGen','Agilent','Affy'),
+                    col=c('red','green','blue'),pch=20)
+     }
+     points(rownames(tvScoresSplit[[ii]]),
+            tvScoresSplit[[ii]][,1],col='blue',
+            pch=20, cex=2, type="p")
+     points(rownames(tvScoresSplit[[ii]]),
+            tvScoresSplit[[ii]][,2],col='red',
+            pch=20, cex=2, type="l")
+     points(rownames(tvScoresSplit[[ii]]),
+            tvScoresSplit[[ii]][,2],col='red',
```

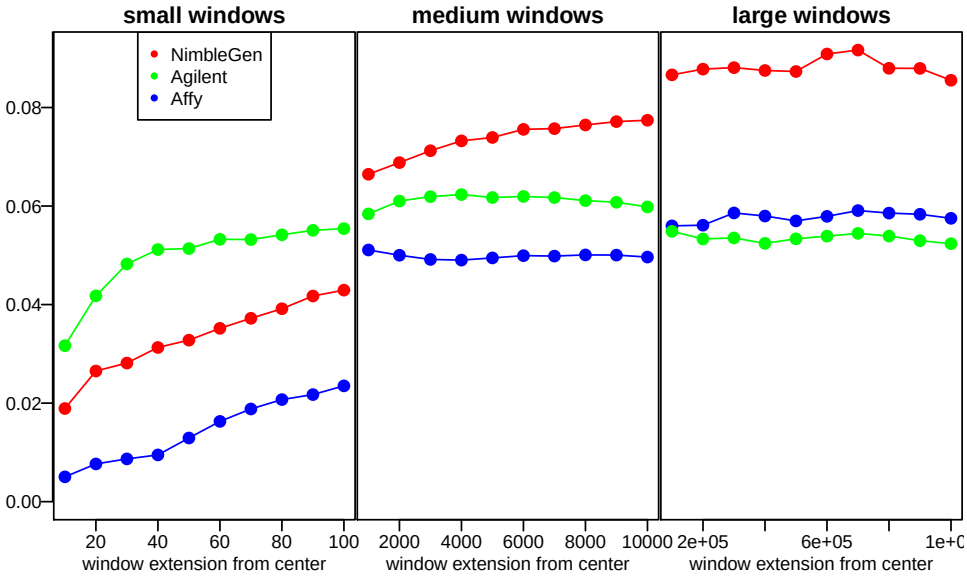

Figure 1: Text describing the figure

```
+             pch=20, cex=2, type="p")
+     points(rownames(tvScoresSplit[[ii]]),tvScoresSplit[[ii]][,3],
+             pch=20, cex=2,
+             col='green', type="l")
+     points(rownames(tvScoresSplit[[ii]]),tvScoresSplit[[ii]][,3],
+             pch=20, cex=2,
+             col='green', type="p")
+
+ }
```

Computing the GC-adjusted M values. We could get rid of this section if we return a list from the original call.

use optimal window sizes to perform correction

```
>     ## We will do correction using two windows for each platform. The window sizes are chosen
>     ## baed upon the tv scores generated in the last section (we choose windows with relatively
>     ## large tv scores-the theory being that windows with larger tv scores are better for
>     ## correction).
> nimcM1 <- gcCorrect(matrix(nimMs),nimchr,nimstarts,samplechr,nodes,increms=6e5,maxwins=6e5,build='hg18
> nimcM2 <- gcCorrect(nimcM1,nimchr,nimstarts,samplechr,nodes,increms=9e3,maxwins=9e3,build='hg18')
> agcM1 <- gcCorrect(matrix(agMs),agchr,agstarts,samplechr,nodes,increms=7e5,maxwins=7e5,build='hg18')
> agcM2 <- gcCorrect(agcM1,agchr,agstarts,samplechr,nodes,increms=4e3,maxwins=4e3,build='hg18')
> afcM1 <- gcCorrect(matrix(afMs),afchr,afstarts,samplechr,nodes,increms=1e3,maxwins=1e3,build='hg18')
> afcM2 <- gcCorrect(afcM1,afchr,afstarts,samplechr,nodes,increms=3e5,maxwins=3e5,build='hg18')

> wave.df <- data.frame(position=c(rep(nimstarts/1e6, 2),rep(agstarts/1e6, 2),rep(afstarts/1e6,2)),
+                       r=c(nimMs, nimcM2,agMs,agcM2,afMs,afcM2),
```

```
+                       platform=rep(c(rep("Nimblegen", 2),
+                       rep("Agilent", 2),
+                       rep("Affymetrix", 2)), c(length(nimMs),
+                                                length(nimcM2),
+                                                length(agMs),
+                                                length(agcM2),
+                                                length(afMs),
+                                                length(afcM2))),
+                       wave.corr=c(rep("raw", length(nimMs)),
+                       rep("ArrayTV", length(nimcM2)),
+                       rep("raw", length(agMs)),
+                       rep("ArrayTV", length(agcM2)),
+                       rep("raw", length(afMs)),
+                       rep("ArrayTV", length(afcM2))))
> 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="log R ratio", xlab="position (Mb) on chr15",
+             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

Use CBS to segment corrected and uncorrected signal within a region containing a known copy number amplification (expected copy number 2.5).

```
> if(require("DNAcopy")){
+         cbs.out <- data.frame()
+         cbs.segs <- list()
+         platforms <- c("Nimblegen", "Agilent", "Affymetrix")
+         correction <- c("raw", "ArrayTV")
+         Ms <- list(nimblegen[, "M"], nimcM2,
+                    agilent[, "M"], agcM2,
```

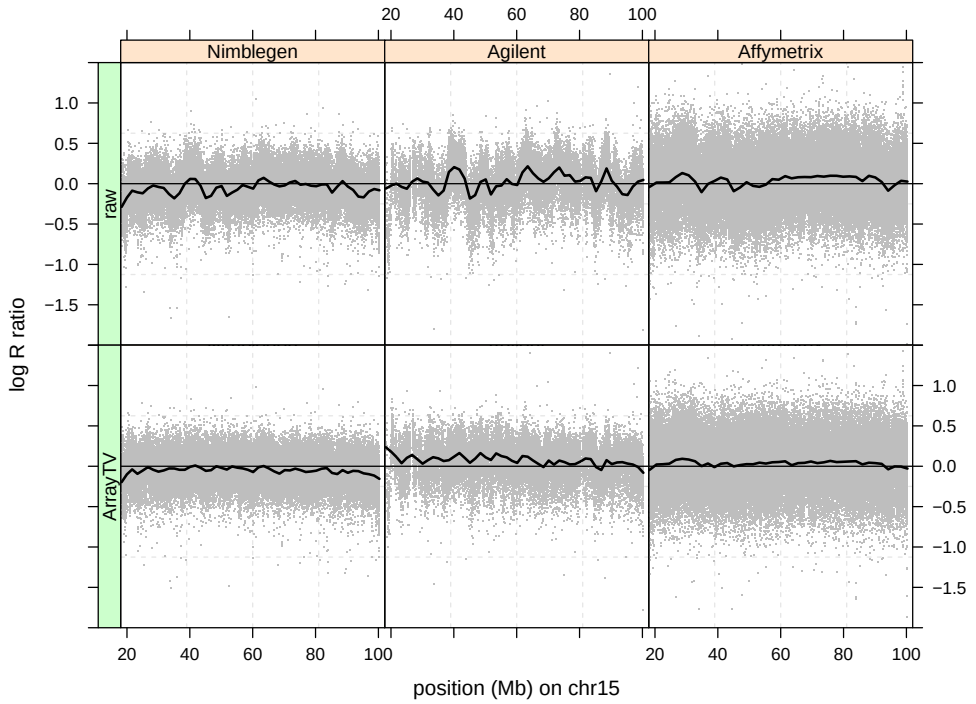

Figure 2: Comparison of log ratios before and after `ArrayTV` wave correction (rows) for three popular array platforms (columns).

```
+                    affymetrix[, "M"], afcM2)
+         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="_"))
+                         ##unique(wave.df$wave.corr)[ii])
+                         cbs.segs[[k]] <- segment(cna1, verbose=1)
+                 }
+         }
+         cbs.out <- do.call("rbind", lapply(cbs.segs, "[[", "output"))
+ }

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

Following is a big code chunk that is complex. Need to break it up and explain what you are doing.

```
> ids <- cbs.out$ID
> platforms <- gsub("_raw", "", ids)
> platforms <- gsub("_ArrayTV", "", platforms)
> correction <- gsub("Nimblegen_", "", ids)
> correction <- gsub("Affymetrix_", "", correction)
> correction <- gsub("Agilent_", "", correction)
> cbs.segs <- GRanges(rep("chr14", nrow(cbs.out)),
+                     IRanges(cbs.out$loc.start, cbs.out$loc.end),
+                     numMarkers=cbs.out$num.mark,
+                     seg.mean = cbs.out$seg.mean,
+                     platform = platforms,
+                     correction=correction)
> cbs.segs.raw <- cbs.segs[cbs.segs$correction=="raw", ]
> cbs.segs.arrayTV <- cbs.segs[cbs.segs$correction=="ArrayTV", ]
> marker.data <- GRanges(rep("chr14", nrow(wave.df)),
+                     IRanges(wave.df$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$correction=="ArrayTV",]
> spikeIn <- IRanges(start=45396506,end=45537976)
> spikeinStart <- 45396506; spikeinEnd <- 45537976
> olaps1 <- findOverlaps(marker.raw, cbs.segs.raw, select="first")
> elementMetadata(marker.raw)$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$seg.mean[olaps2]
> values(marker.raw)$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)$spikeIn[index2] <- NA
> rawd <- as.data.frame(marker.raw)
> atvd <- as.data.frame(marker.arrayTV)
> rawd <- rbind(rawd, atvd)
> ##rawd <- as.data.frame(raw.data)
> rawd$platform <- factor(rawd$platform, levels=c("Nimblegen",
+                                                 "Agilent", "Affymetrix"))
> rawd$correction <- factor(rawd$correction, levels=c("raw", "ArrayTV"))
```

View CBS lines through the spike in (expected copy number 2.5). Black points represent signal intensities for each probe. Red lines represent segmentation of the intensities (using the circular binary segmentaion algorithm) and blue lines represent a region of known copy number 2.5 (a subtle amplification). 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="log R ratio",
+               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)
```

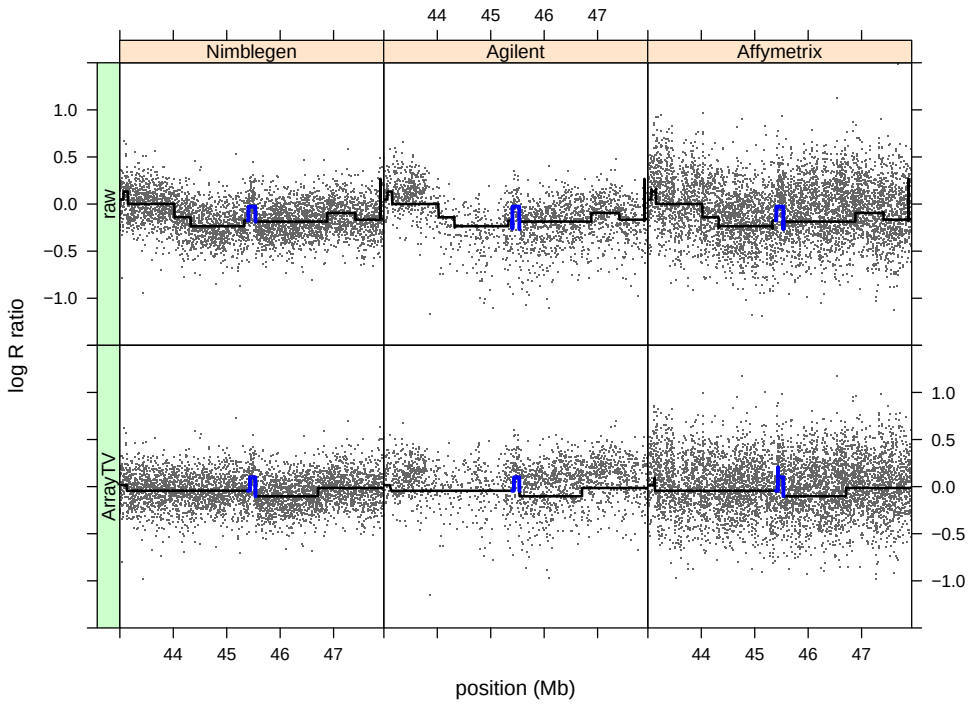

Figure 3: Some text describing the figure

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 1.1.28, BiocGenerics 0.7.2, BiocInstaller 1.11.1, Biostrings 2.29.3, BSgenome 1.29.0, BSgenome.Hsapiens.UCSC.hg18 1.3.19, DNAcopy 1.35.0, GenomicRanges 1.13.14, IRanges 1.19.8, 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 2.21.2, bit 1.1-10, codetools 0.2-8, compiler 3.1.0, ff 2.2-11, foreach 1.4.0, grid 3.1.0, iterators 1.0.6, stats4 3.1.0, tools 3.1.0, zlibbioc 1.7.0

References

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.