Correcting gene specific dye bias

Philip Lijnzaad[1] and Thanasis Margaritis[2]

September 2, 2010

1. <p.lijnzaad@umcutrecht.nl>
2. <a.margaritis@umcutrecht.nl>
Holstege Laboratory
Dept. of Physiological Chemistry
University Medical Center (UMC)
Universiteitsweg 100
3584 CG Utrecht
The Netherlands

This is the complete vignette; the one produced with R CMD build is a cut-down version.

Contents

1 Overview

This document gives a brief tutorial on how to use the `dyebias` package to correct for gene-specific dye bias of two-colour microarray data using the GASSCO method by Margaritis et al. (2009).

A well-known artefact of two-colour microarrays is the intensity-dependent dye bias, which is usually adequately addressed by using LOESS normalization. However, gene-specific dye bias is different, and persists after LOESS normalization. It is also not sufficient to use dye swaps, as this firstly wastes statistical power, but more importantly, may leave residual bias since dye-swaps need not have the same degree of dye bias.

Gene-specific dye bias is ubiquitous, and often quite large. The GASSCO method implemented in the `dyebias` package efficiently solves the problem in a general and robust way. The principle behind the method – which can equally well be called a normalization method – is very simple. We observed that the total dye bias varies not only with each gene (or, more precisely, each probe or reporter), but also with each hybridization, in an apparently linear fashion. The formula is simply

$\beta_{ij} = F_j * \gamma_i$

where β_{ij} is the total dye bias of reporter i in hybridization j; F_j is the estimated *slide bias* (more precisely: the hybridization bias), and γ_i is the so-called *intrinsic gene specific dye bias*, or briefly *iGSDB* (again: *gene* is a misnomer; *probe* or *reporter* is more precise).

It turns out that the *iGSDB* depends on the protocols and somewhat on the average expression levels, but the largest factor is the reporter sequence. The slide bias depends largely on the labeling percentage (Margaritis et al., 2009). Although the *iGSDB*s can be predicted from the reporter sequence (see Margaritis et al. (2009), Supplemental Discussion), it is easier and more precise to estimate it from the data. All that is needed is either a number of self-self hybridizations, or a number of experiments that have been done in dye-swap (also known as dye-flip or fluor-flip). If needed, estimated *iGSDB*s can be reused, provided the protocols have not changed. The genes having the strongest *iGSDB*s are used to estimate the slide bias.

The steps to perform dye bias correction are, then,

1. estimate the intrinsinc gene specific dye bias using a number of self-self or dye-swapped slides
2. for each slide, estimate the slide bias, and apply the correction

The `dyebias` package provides functions to do both, and also has some plotting routines to judge the correction. The package has been tested on a wide variety of public data (see Margaritis et al. (2009)), and has been in active production in our own lab since over a year. We appreciate all feedback.

2 Installing

If you are reading this document and have not yet installed any software on your computer, visit `http://bioconductor.org` and follow the instructions for installing `R` and Bioconductor. Once you have installed both `R` and Bioconductor, install the `dyebias` package with

```
source("http://bioconductor.org/biocLite.R")
biocLite("dyebias")
```

Alternatively, under Windows, select `Packages` from the menu and click on `Install package(s) from Bioconductor...`, and select `dyebias` from the pop–up. Under Linux/Unix you can use the

usual command `R CMD INSTALL` or set the option `CRAN` to your nearest mirror site and use the command `install.packages` from within an R session. The latest version of the `dyebias` package can always be found at `http://www.holstegelab.nl/publications/margaritis_lijnzaad`.

Note: if you run `R CMD check` on this package, it will complain with `dyebias.apply.correction: no visible global function definition for 'maR<-'` (nor for `'maG<-'`). You can safely ignore this message.

3 Introduction

For simple examples of how the `dyebias` package can be used to correct dyebias, just run the `example()`s for any of the functions in the package. In particular, try `example(dyebias.rgplot)`, `example(dyebias.boxplot)` and `example(dyebias.trendplot)`. They make use of our own validation data, which is provided by the `dyebiasexamples` package, which can be installed through Bioconductor similarly. The current document presents two more case studies.

4 Tuteja et al. (2008)

The first example uses data from Tuteja et al. (2008), a genome-wide location analysis of the transcription factor Foxa2, which is important in liver development. The ChIP on chip data comes from five self-spotted mouse cDNA slides.

4.1 Reading the data

The data is deposited in Array Express (Parkinson et al., 2007) under accession `M-TAB-32`, but for convenience, the data is provided through the `dyebiasexamples` package. We need the `marray` package (Yang et al., 2007) for the data model, `convert` for converting, and of course `dyebias` and `dyebiasexamples`:

```
> library(marray)
> library(dyebias)
> library(dyebiasexamples)
> library(convert)
> options(stringsAsFactors = FALSE)
```

The Tuteja data itself is not in the `data` directory, since it doesn't adhere to the `R` standards. Instead, it was put in the `extradata` directory. Find it:

```
> datadir <- system.file("extradata", package = "dyebiasexamples")
> datadir

[1] "/usr/local/lib/R/site-library/dyebiasexamples/extradata"
```

The sample file contains descriptions of the hybridizations; read it and convert to a proper `marrayInfo` object:

```
> sample.file <- file.path(datadir, "Tuteja-samples.txt")
> targets <- read.marrayInfo(sample.file)
> summary(targets)

Object of class marrayInfo.

  maLabels slide                          filename   Cy3   Cy5
1        1     1 E-MTAB-32-raw-data-1641029399.txt input FoxA2
2        2     2 E-MTAB-32-raw-data-1641029407.txt input FoxA2
3        3     3 E-MTAB-32-raw-data-1641029415.txt FoxA2 input
4        4     4 E-MTAB-32-raw-data-1641029423.txt FoxA2 input
5        5     5 E-MTAB-32-raw-data-1641029431.txt input FoxA2

Number of labels:  5
Dimensions of maInfo matrix:  5  rows by  4  columns

Notes:
 /usr/local/lib/R/site-library/dyebiasexamples/extradata/Tuteja-samples.txt
```

It shows that we have five hybridizations, all of them FoxA2-enriched versus input material. Three of them have the dyes in the "forward" direction, two in the "reverse" direction. In other words, the design is not balanced; more on this below. First prepare the layout information (this was gleaned from the original submission):

```
> layout <- read.marrayLayout(fname = NULL, ngr = 12, ngc = 4,
+     nsr = 22, nsc = 19)
> n.spots <- maNspots(layout)
> summary(layout)

Array layout:         Object of class marrayLayout.

Total number of spots:                        20064
Dimensions of grid matrix:                12 rows by 4 cols
Dimensions of spot matrices:                22 rows by 19 cols

Currently working with a subset of 20064spots.

Control spots:


Notes on layout:
 No Input File
```

Read the gene information from the first of the data files. The only thing really needed by the `dyebias` package is the `reporterId` column for the `maInfo(maGnames(data))`, but it doesn't hurt to include `control.type` and `reporter.group`. The latter is used to set the `maControls`. These are not needed, but `marray` complains otherwise, so we might as well supply the information.

```
> first.file <- file.path(datadir, maInfo(targets)$filename[1])
> first.file
```

```
[1] "/usr/local/lib/R/site-library/dyebiasexamples/extradata/E-MTAB-32-raw-data-1641029399.txt"
```

```
> gnames <- read.marrayInfo(first.file, info.id = c(7, 8, 10),
+     labels = 10, skip = 0)
> names(maInfo(gnames)) <- c("control.type", "reporter.group",
+     "reporterId")
> maInfo(gnames)$reporterId <- as.character(maInfo(gnames)$reporterId)
> controls <- maInfo(gnames)$reporter.group
> controls[controls == "Experimental"] <- "gene"
> maControls(layout) <- as.factor(controls)
> summary(gnames)
```

```
Object of class marrayInfo.

   maLabels control.type reporter.group reporterId
1   P101020               Experimental    P101020
2   P101770               Experimental    P101770
3   P102903               Experimental    P102903
4   P103743               Experimental    P103743
5   P104397               Experimental    P104397
6   P104943               Experimental    P104943
7   P105309               Experimental    P105309
8   P106233               Experimental    P106233
9   P203560               Experimental    P203560
10  P202765               Experimental    P202765
...

Number of labels:  20064
Dimensions of maInfo matrix:  20064  rows by  3  columns

Notes:
 /usr/local/lib/R/site-library/dyebiasexamples/extradata/E-MTAB-32-raw-data-1641029399.txt
```

Next, we read the data. In the examples in this document, we assume that the public data is properly normalized; that is really the starting point for GASSCO. If the data is not (properly) normalized, results will be suboptimal (as we will see). In the Tuteja data set both 'raw' and normalized data are available. This is good, because the raw data has foreground as well as background intensities; the latter allow us to better judge the quality of the spots. We therefore read both, starting with the raw data:

```
> data.raw <- read.GenePix(fnames = maInfo(targets)$filename, path = datadir,
+     name.Gf = "GenePix:F532 Median", name.Gb = "GenePix:B532 Median",
+     name.Rf = "GenePix:F635 Median", name.Rb = "GenePix:B635 Median",
+     name.W = "GenePix:Flags", layout = layout, gnames = gnames,
+     targets = targets, skip = 0, sep = "\t", quote = "\"", DEBUG = TRUE)
```

```
[1] "E-MTAB-32-raw-data-1641029399.txt" "E-MTAB-32-raw-data-1641029407.txt" "E-MTAB-32-raw-data

Control    gene
   2174   17890
Setting up controls status ... done
 Calling read.marrayRaw ...
Reading ...  /usr/local/lib/R/site-library/dyebiasexamples/extradata/E-MTAB-32-raw-data-164102
Reading ...  /usr/local/lib/R/site-library/dyebiasexamples/extradata/E-MTAB-32-raw-data-164102
Reading ...  /usr/local/lib/R/site-library/dyebiasexamples/extradata/E-MTAB-32-raw-data-164102
Reading ...  /usr/local/lib/R/site-library/dyebiasexamples/extradata/E-MTAB-32-raw-data-164102
Reading ...  /usr/local/lib/R/site-library/dyebiasexamples/extradata/E-MTAB-32-raw-data-164102
```

The weights in the file use a different convention, so convert them to the `marray` convention, which is `weight == 0` for bad spots, `weight == 1` for good spots. (Bad spots are never used as estimator genes, and you could also decide not to dye bias-correct them). Also attach the layout to the `data.raw` object, and show what we have:

```
> maW(data.raw)[maW(data.raw) == 0] <- 1
> maW(data.raw)[maW(data.raw) < 0] <- 0
> maLayout(data.raw) <- layout
> summary(data.raw)

Pre-normalization intensity data:         Object of class marrayRaw.

Number of arrays:         5 arrays.

A) Layout of spots on the array:
Array layout:         Object of class marrayLayout.

Total number of spots:                        20064
Dimensions of grid matrix:                12 rows by 4 cols
Dimensions of spot matrices:                22 rows by 19 cols

Currently working with a subset of 20064spots.

Control spots:
There are   2 types of controls :

Control    gene
   2174   17890


Notes on layout:
 No Input File

B) Samples hybridized to the array:
```

```
Object of class marrayInfo.

  maLabels slide                          filename   Cy3   Cy5
1        1     1 E-MTAB-32-raw-data-1641029399.txt input FoxA2
2        2     2 E-MTAB-32-raw-data-1641029407.txt input FoxA2
3        3     3 E-MTAB-32-raw-data-1641029415.txt FoxA2 input
4        4     4 E-MTAB-32-raw-data-1641029423.txt FoxA2 input
5        5     5 E-MTAB-32-raw-data-1641029431.txt input FoxA2

Number of labels:  5
Dimensions of maInfo matrix:  5  rows by  4  columns

Notes:
 /usr/local/lib/R/site-library/dyebiasexamples/extradata/Tuteja-samples.txt

C) Summary statistics for log-ratio distribution:
                                                                                          Min.
/usr/local/lib/R/site-library/dyebiasexamples/extradata/E-MTAB-32-raw-data-1641029399.txt -7.12
/usr/local/lib/R/site-library/dyebiasexamples/extradata/E-MTAB-32-raw-data-1641029407.txt -7.00
/usr/local/lib/R/site-library/dyebiasexamples/extradata/E-MTAB-32-raw-data-1641029415.txt -6.86
/usr/local/lib/R/site-library/dyebiasexamples/extradata/E-MTAB-32-raw-data-1641029423.txt -6.99
/usr/local/lib/R/site-library/dyebiasexamples/extradata/E-MTAB-32-raw-data-1641029431.txt -8.09

D) Notes on intensity data:
GenePix Data
```

Let's now read the normalized data. As a short-cut, we will use `data.raw` as a template (using `as()` from the `convert` package), and then only replace the actual values.

```
> normalized.data.file <- file.path(datadir, "E-MTAB-32-processed-data-1641029599.txt")
> data <- read.table(normalized.data.file, sep = "\t", as.is = T,
+     header = T, skip = 1)
> names(data) <- c("reporterId", "X13685041", "X13685040", "X13685153",
+     "X13685151", "X13685042")
> data$reporterId <- sub(pattern = "ebi.ac.uk:MIAMExpress:Reporter:A-MEXP-1165\\.(P[0-9]+) .*",
+     replacement = "\\1", x = data$reporterId, perl = T, fixed = F)
> data$reporterId <- sub(pattern = "ebi.ac.uk:MIAMExpress:Reporter:A-MEXP-1165\\.Blank.*",
+     replacement = "Blank", x = data$reporterId, perl = T, fixed = F)
> data.norm <- as(data.raw, "marrayNorm")
> m <- as.matrix(data[, c(2:6)])
> n <- as.numeric(m)
> dim(n) <- dim(m)
> maM(data.norm) <- n
```

In most cases we have seen, authors have deposited their data such that the dye swaps have been undone so that the slides all seem to be in the "forward" direction. (The reason is that you can

then more easily compare the treatment vs. reference behaviour of a set of slides, regardless of dye orientation.

However, here we are interested only in the dye effect itself, that is, in the effect of $log_2(Cy5/Cy3)$, *not* in the difference of FoxA2-enriched over input material. We therefore have to reswap the undone dye swaps. [1]

```
> maM(data.norm)[, c(3, 4)] <- -1 * maM(data.norm)[, c(3, 4)]
```

A, the average intensity of both channels, is not yet set, so compute it here:

```
> maA(data.norm) <- log2(maRf(data.raw) * maGf(data.raw))/2
```

Note that the code does not really need `A`; the correction is based solely on `M`. The `A` is not always available; some labs publish only the `M`-values. In such cases, it is fine to "invent" an `A`, e.g. by sampling from a uniform distribution between 2 and 15 (which is the typical domain of `A`). The advantage of such fake data is that the plotting routines in the `dyebias` package continue to work.

4.2 Estimating the *iGSDB*

We now have a full-fledged `marrayNorm` object that we can use to estimate the iGSDBs from.

The design is unbalanced, which is actually uncommon; most designs are (or should be) balanced. If the design is unbalanced, the `dyebias.estimate.iGSDBs` function uses `limma` (Smyth, 2005) to obtain estimates for the intrinsic gene specific dye biases. In the current data set, the `reference` is the sample called `input` (see `maInfo(maTargets(data.norm))`). The call is therefore:

```
> iGSDBs.estimated <- dyebias.estimate.iGSDBs(data.norm, is.balanced = FALSE,
+     reference = "input", verbose = TRUE)

[1] "Estimating the intrinsic gene-specific dyebiases ..."
[1] "Found 5 slides containing 20064 spots"
[1] "Found 20064 spots, 19297 of them unique. Averaging replicate spots ...\n"
[1] "Done averaging replicate spots"
[1] "Found common-reference design"
Found unique target names:
 FoxA2 input
[1] "Design\n"
  FoxA2 Dye
1     1   1
2     1   1
3    -1   1
4    -1   1
5     1   1
```

[1] Whether the columns have been deposited the right way around is often not clear from the description. Sometimes it can be inferred from the general behaviour of the data. E.g., in the case of knock-down data, the knocked-down gene should be down in one hybridization, up in the other; in ChIP data, the ChIP-enriched sample generally should go up relative to the input sample and is therefore be visible, in an MA-plot, as a cloud of points with a flattish bottom and a bulging top.

```
[1] "fitting linear model ..."
[1] "intrinsic GSDB's:\n"
      Min.    1st Qu.     Median       Mean    3rd Qu.       Max.       NA's
 -0.968000  -0.067000   0.004000  -0.006515   0.064000   0.852000 100.000000
[1] "Done estimating the intrinsic gene-specific dyebiases."
```

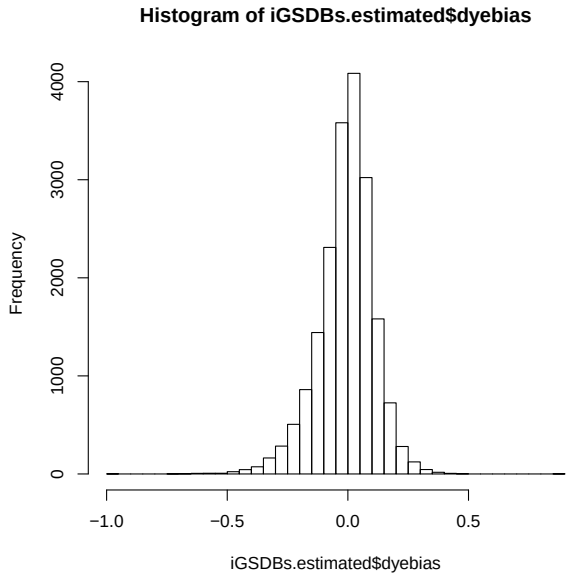
```
> summary(iGSDBs.estimated)
```

```
  reporterId           dyebias                A             p.value
 Length:19297       Min.   : -0.968000   Min.   : 6.580   Min.   :  0.00000
 Class :character   1st Qu.: -0.067000   1st Qu.: 7.780   1st Qu.:  0.06153
 Mode  :character   Median :  0.004000   Median : 8.510   Median :  0.24282
                    Mean   : -0.006515   Mean   : 8.704   Mean   :  0.33240
                    3rd Qu.:  0.064000   3rd Qu.: 9.500   3rd Qu.:  0.56491
                    Max.   :  0.852000   Max.   :14.930   Max.   :  1.00000
                    NA's   :100.000000                    NA's   :100.00000
```

This took a little while, mostly be because a full Limma model has to be run for this unbalanced design. (We ignore its p-values, since we have no use for them). If the design is balanced, the estimate is done by simply averaging. This gives identical results (and no p-values), but is much faster.

Let's have a quick look at the distribution of the iGSDBs:

```
> hist(iGSDBs.estimated$dyebias, breaks = 50)
```



In our own lab, the iGSDBs generally have a distribution that is roughly symmetrical, with a mean slightly below zero, and a standard deviation of around 0.2. The data shown here shows is fairly similar.

4.3 Applying the dye bias correction

Before applying the correction, we need to choose which genes (probes/reporters!) are good enough to base the slide bias estimate on. We call them the estimator genes. Generally, we discard reporters that are likely to have a high biological variability. Generally, reporters not representing genes, and those corresponding to mitochondrial genes, transposons, cross-hybridizing and non-unique oligo sequences are therefore discarded. In the current case, we know nothing about the reporters, so we only insist that they correspond to genes; that group is called `Experimental`:

```
> estimator.subset <- (maInfo(maGnames(data.norm))$reporter.group ==
+     "Experimental")
> summary(estimator.subset)

   Mode   FALSE    TRUE    NA's
logical    2174   17890       0
```

A convenience function is provided to get an index for the spots that are trustworthy estimator genes. The current data set does have background signals, so we specify `use.background=TRUE`. (If background signals would not have been available, we should have specified `use.background=FALSE`, which causes the minimum foreground signal to be used as a proxy for the background signal).
We now specify which spots we would like to dye bias-correct. Generally it is best to correct all spots. However, we have noticed that for signals that are too high or too low, over-correction can occur. Signals that are too high are due to saturation of the scanner, whereas those that are too low may be too noisy due to differences in background signal. We want to be a bit more selective, only correcting genes where we are certain that we have a trustworthy signal. We therefore insist that the weight of the spot is 1, and that they lie in the right range. Here, we choose a signal-to-noise ratio (here defined as foreground over background signal) ≥ 1.5 and an average expression `A` ≤ 15.

```
> application.subset <- ((maW(data.norm) == 1) & dyebias.application.subset(data.raw = data.raw
+     min.SNR = 1.5, maxA = 15, use.background = TRUE))
> summary(as.vector(application.subset))

   Mode   FALSE    TRUE    NA's
logical   22927   77393       0
```

We're now set to apply the correction, simply as:

```
> correction <- dyebias.apply.correction(data.norm = data.norm,
+     iGSDBs = iGSDBs.estimated, estimator.subset = estimator.subset,
+     application.subset = application.subset, verbose = TRUE)

[1] "Average log2(signal) for estimator genes: min=6.58, max=14.93\nExcluding those outside 7.8
[1] "Top 5 % reporters (20064 spots; 17890 of them candidates, 9037 of them in the right range,
[1] "Dyebias correcting slide 1 (1)...\n"
[1] "Estimated slide factor\n: green based: 0.453385, red based: 0.725926\nAverage correction 0
[1] "Excluding 5015 out of 20064 spots from dyebias correction\n"
[1] "variance ratio (i.e. corrected/uncorrected): 0.853\nOveral reduction in variance is 14.7 %
```

```
[1] "Done correcting slide 1 (1)\n"
[1] "Dyebias correcting slide 2 (2)...\n"
[1] "Estimated slide factor\n: green based: 0.74759, red based: 1.16495\nAverage correction 0.9
[1] "Excluding 4356 out of 20064 spots from dyebias correction\n"
[1] "variance ratio (i.e. corrected/uncorrected): 0.803\nOveral reduction in variance is 19.7 %
[1] "Done correcting slide 2 (2)\n"
[1] "Dyebias correcting slide 3 (3)...\n"
[1] "Estimated slide factor\n: green based: 1.97027, red based: 1.09978\nAverage correction 1.4
[1] "Excluding 4392 out of 20064 spots from dyebias correction\n"
[1] "variance ratio (i.e. corrected/uncorrected): 0.513\nOveral reduction in variance is 48.7 %
[1] "Done correcting slide 3 (3)\n"
[1] "Dyebias correcting slide 4 (4)...\n"
[1] "Estimated slide factor\n: green based: 1.02166, red based: 1.00505\nAverage correction 1.0
[1] "Excluding 4019 out of 20064 spots from dyebias correction\n"
[1] "variance ratio (i.e. corrected/uncorrected): 0.612\nOveral reduction in variance is 38.8 %
[1] "Done correcting slide 4 (4)\n"
[1] "Dyebias correcting slide 5 (5)...\n"
[1] "Estimated slide factor\n: green based: 0.537208, red based: 0.866387\nAverage correction 0
[1] "Excluding 5145 out of 20064 spots from dyebias correction\n"
[1] "variance ratio (i.e. corrected/uncorrected): 0.869\nOveral reduction in variance is 13.1 %
[1] "Done correcting slide 5 (5)\n"
[1] "Done dyebias-correcting the slides. Summary of the variance-reduction percentages:\n"
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  13.08   14.66   19.72   26.99   38.77   48.70
```

The `correction` object contains the complete results. It is a `list` with 4 members: `data.corrected`, the corrected data in `marrayNorm` format; `estimators`, which contains the estimators and is used by the plotting routines; `summary`, a `data.frame` with statistics concerning the dye bias correction, and lastly (for convenience) `data.uncorrected`, the input data. For the summary, the most useful ones are probably `avg.correction, var.ratio, reduction.perc` and `p.value`:

```
> correction$summary[, c("slide", "avg.correction", "var.ratio",
+     "reduction.perc", "p.value")]

  slide avg.correction var.ratio reduction.perc       p.value
1     1      0.5958807 0.8533677       14.66323  3.316962e-22
2     2      0.9069777 0.8027584       19.72416  8.543756e-43
3     3      1.4247242 0.5130049       48.69951  0.000000e+00
4     4      1.0069697 0.6123070       38.76930 1.539719e-208
5     5      0.6597780 0.8692236       13.07764  1.467025e-17
```

`slide` (or alternatively `file`) identifies the slide; `avg.correction` is the slide bias; `var.ratio` shows the reduction in the variance of `M`) that was achieved by the correction; `reduction.perc` showing the same, but as a percentage. Lastly, `p.value` indicates the significance of the reduction in variance (H_0: variances before and after correction are equal).

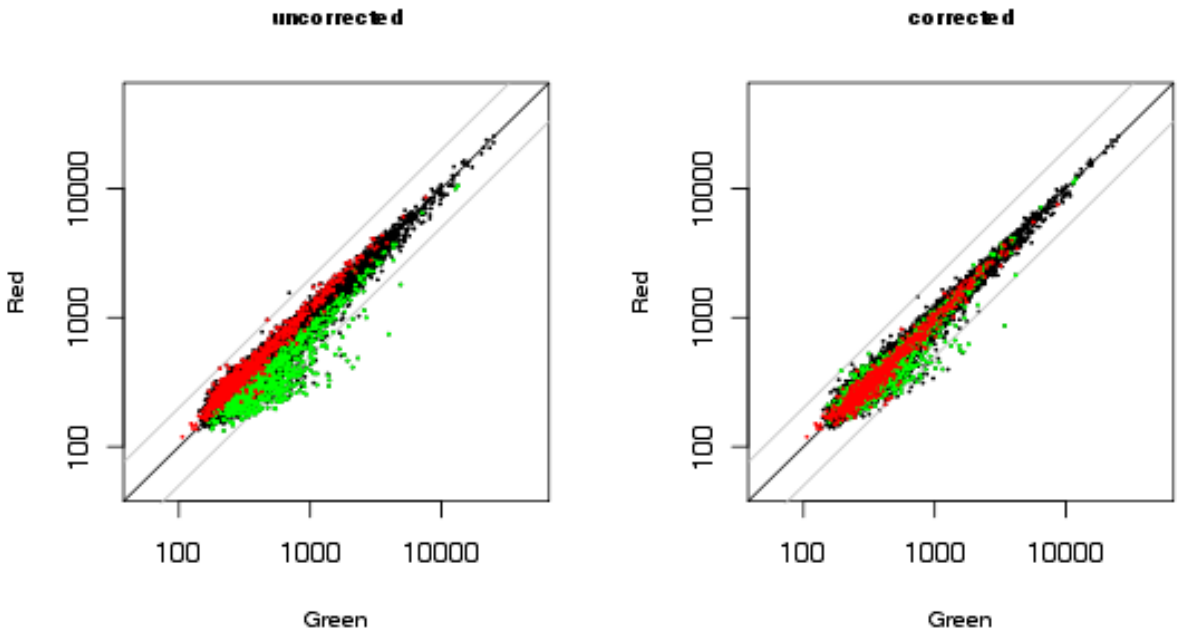
It can be seen that slide 3 had the strongest dye bias. Its variance of M is hugely reduced, by 50%.

4.4 Plotting the data

Let us look at how the correction worked out for one particular slide of this set, say number 3, the one with the strongest slide bias. We first plot the uncorrected data, then the corrected data. In both cases, we colour the strongest 5% green-biased and red-biased spots accordingly (that is why the `iGSDBs`-argument is required).

There will be an overlap between the estimator genes and those now coloured red and green. However, the estimator genes only come from the "middle" of the data set (as specified by the default value of the `minmaxA.perc`-argument; see the documentation).

```
> layout(matrix(1:2, nrow = 1, ncol = 2))
> dyebias.rgplot(data = data.norm, slide = 3, iGSDBs = iGSDBs.estimated,
+     application.subset = application.subset, main = "uncorrected",
+     cex = 0.2, cex.lab = 0.8, cex.main = 0.8, output = NULL)
> dyebias.rgplot(data = correction$data.corrected, slide = 3, iGSDBs = iGSDBs.estimated,
+     application.subset = application.subset, main = "corrected",
+     cex = 0.2, cex.lab = 0.8, cex.main = 0.8, output = NULL)
```

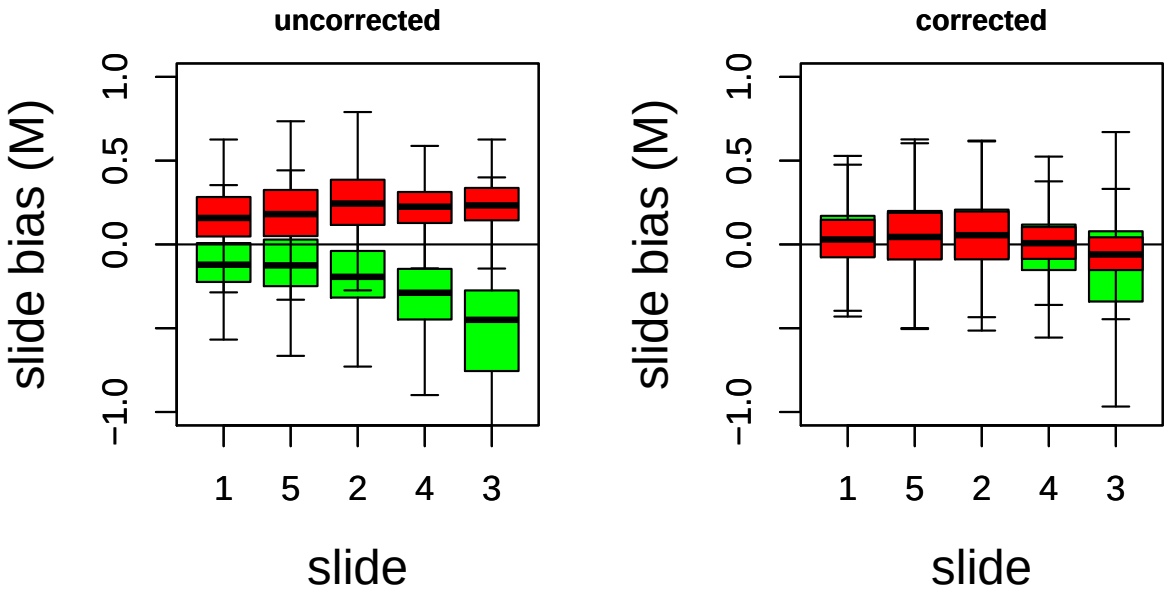


In the uncorrected slide, there is a clear separation between the genes with a strong red bias, and those with a strong green bias. After correction, this separation has disappeared, and the overal spread around the diagonal is much reduced. As can be seen from from the off-diagonal lines (representing a two-fold change), some 20 spots appear to have been down by more than two-fold down just as a result of gene-specific dye bias.

How did the procedure perform for all slides? This is best seen using the `dyebias.boxplot` function. It is called twice, once with the uncorrected, once with the corrected data. For clarity,

we prefer to order the slides by increasing slide bias (of the uncorrected data), hence the `order` argument:

```
> layout(matrix(1:2, nrow = 1, ncol = 2))
> order = dyebias.boxplot(data = data.norm, iGSDBs = iGSDBs.estimated,
+     order = NULL, ylim = c(-1, 1), application.subset = application.subset,
+     main = "uncorrected", cex = 0.2, cex.lab = 0.8, cex.main = 0.8,
+     output = NULL)
> order = dyebias.boxplot(data = correction$data.corrected, iGSDBs = iGSDBs.estimated,
+     order = order, ylim = c(-1, 1), application.subset = application.subset,
+     main = "corrected", cex = 0.2, cex.lab = 0.8, cex.main = 0.8,
+     output = NULL)
```



If the red and green boxes coincide, this is a sign that the bias was removed successfully, and without introducing other biases. This is clearly the case here. Although in general there is no correlation between slide bias and residual bias after correction, slide number 3 has both the largests slide bias and the largest residual bias. One could consider throwing this hybridization out.

In the above, we have restricted the dyebias correction and the plots to spots that had good quality, by insisting on a weight of 1, good signal-to-noise and no saturation. If you would plot all spots (by leaving out the `application.subset` argument), you would get a fairly large residual dye bias. The reason is that this data set has a high proportion of spots that we chose not to correct: around 15% is due to low quality spots, and around 25% due to too low or too high intensity. The solution is to play with the `application.subset` to get more spots corrected; this is left as an exercise to the reader.

5 Chen et al. (2007)

The next example is based on genome-wide location data of Orc6, which plays a role in DNA replication initiation (Chen et al., 2007). The data is available from NCBI's Gene Expression Omnibus (Barrett et al., 2007) under accession `GSE9318`. It consists of two sets of Agilent slides: 4 done on platform `GPL3499`, and 9 slides done on platform `GPL5991`. Only the latter set has proper dye swaps, and these will be used to estimate the intrinsic gene-specific dye bias (iGSDB). These estimates are subsequently used to correct all 9 `GPL5991` slides.

5.1 Reading the data

We will try and download the data directly from GEO using the `GEOquery` (Davis and Meltzer, 2007)). The downloading and the subsequent parsing takes a long while, so we'll first see if we already have the `.RData` file (derived from the `.soft` file), if not, prompt for downloading it to the current directory, and then parse the file and save it as an `R` data dump. We need the full `.soft`-file, since we want to have all the data, not just the `M`-values.

```
> library(GEOquery)
> library(marray)
> library(limma)
> library(dyebias)
> library(dyebiasexamples)
> options(stringsAsFactors = FALSE)
> gse.id <- "GSE9318"
> dir <- getwd()
> if (!file.exists(dir)) {
+     stop(sprintf("could not find directory '%s' (to write GSE9318 data to/from)",
+         dir))
+ }
> file.RData <- sprintf("%s/%s.RData", dir, gse.id)
> if (file.exists(file.RData)) {
+     cat(sprintf("Loading existing data from %s\n", file.RData))
+     load(file.RData)
+ } else {
+     if (interactive()) {
+         if (readline(prompt = sprintf("Could not find file %s.\nDo you want me to download th
+             file.RData)) != "y") {
+             stop("Exiting")
+         }
+     }
+     cat("Downloading .soft file from GEO ...\n")
+     file.soft <- getGEOfile(gse.id, destdir = dir, amount = "full")
+     cat("done\nParsing .soft file ...\n")
+     gse <- getGEO(filename = file.soft)
+     cat(sprintf("done\nSaving to %s ... ", file.RData))
+     save.image(file.RData)
```

```
+     cat("done\n")
+ }
```

```
Loading existing data from /svn/bioconductor.trunk/dyebias/inst/doc/GSE9318.RData
```

This data uses the GEOquery classes, and this has to be converted to `marrayNorm` and `marrayRaw` objects, since that is what the `dyebias` package uses. The `dyebias.geo2marray`-function in the `dyebiasexamples` package is provided for that.

The data set consists of slides done on two platforms. The only hybridizations that were done in dye-swap on the same platform are on platform `GPL5991`. These slides are `GSM237352, GSM237353, GSM237354` and `GSM237355`, and we will use them to estimate the iGSDBs. We subsequently use this estimate to correct all the slides done the `GPL5991` platform[2].

The reporter label is found under the `ProbeName`-column, and probes (reporters) corresponding to genes look like `A_75_P01000003`; this is used for the `gene.selector`-function.

Since the normalized data does not contain `A`-values, we calculate (approximate) them from the raw data. The columns etc. are as indicated. You generally have to be careful with this; use e.g. `Columns((GSMList(gse))[[1]])` to see what they are, and also plot them to be sure about the dye orientation.

```
> slide.ids.est <- c("GSM237352", "GSM237353", "GSM237354", "GSM237355")
> cy3.name <- "label_ch1"
> cy5.name <- "label_ch2"
> gene.selector <- function(table) {
+     grep("^A_75_", as.character(table[["ProbeName"]]))
+ }
> reporterid.name <- "ProbeName"
> M.name <- "VALUE"
> Gf.name <- "Cy3"
> Gb.name <- "Cy3_Background"
> Rf.name <- "Cy5"
> Rb.name <- "Cy3_Background"
> data.raw.est <- dyebias.geo2marray(gse = gse, slide.ids = slide.ids.est,
+     type = "raw", gene.selector = gene.selector, reporterid.name = reporterid.name,
+     cy3.name = cy3.name, cy5.name = cy5.name, Rf.name = Rf.name,
+     Gf.name = Gf.name, Rb.name = Rb.name, Gb.name = Gb.name,
+     )
> data.norm.est <- dyebias.geo2marray(gse = gse, slide.ids = slide.ids.est,
+     type = "norm", gene.selector = gene.selector, reporterid.name = reporterid.name,
+     cy3.name = cy3.name, cy5.name = cy5.name, M.name = M.name)
> maA(data.norm.est) <- log2(maRf(data.raw.est) * maGf(data.raw.est))/2
> maM(data.norm.est)[, c(2, 4)] <- -maM(data.norm.est)[, c(2, 4)]
```

There is one thing that is not quite right with the current data: the contents of the `Cy3`- and `Cy5`-columns are wrong:

[2]The slides done on the other platform (`GPL3499`) in this data set may or may not benefit from dye bias correction using iGSDB estimates from the `GPL5991` slides; this is left as an exercise to the reader.

```
> maInfo(maTargets(data.norm.est))
```

```
           filename Cy3 Cy5    type      description slide
GSM237352 GSM237352 Cy5 Cy3 genomic ORC CHIP in G2/M     1
GSM237353 GSM237353 Cy3 Cy5 genomic ORC CHIP in G2/M     2
GSM237354 GSM237354 Cy5 Cy3 genomic ORC CHIP in G2/M     3
GSM237355 GSM237355 Cy3 Cy5 genomic ORC CHIP in G2/M     4
```

5.2 Estimating the *iGSDB*

It happens not to matter in the current case, because the design is balanced, and therefore, the dyebias is simply the average. However, we'll correct it nonetheless as it is instructive. A bit of sleuthing shows that what is actually needed is the following:

```
> info <- maInfo(maTargets(data.norm.est))
> info$Cy3 <- c("wt", "wt IP", "td", "td IP")
> info$Cy5 <- c("wt IP", "wt", "td IP", "td")
> references <- c("wt", "td")
> maInfo(maTargets(data.norm.est)) <- info
> maInfo(maTargets(data.norm.est))
```

```
           filename   Cy3   Cy5    type      description slide
GSM237352 GSM237352    wt wt IP genomic ORC CHIP in G2/M     1
GSM237353 GSM237353 wt IP    wt genomic ORC CHIP in G2/M     2
GSM237354 GSM237354    td td IP genomic ORC CHIP in G2/M     3
GSM237355 GSM237355 td IP    td genomic ORC CHIP in G2/M     4
```

That is, we have one dye-swapped pair of IP vs. input material for the wild type, and a similar pair for the temperature-sensitive degron mutant. In other words, we have a set of pairwise designs. If `dyebias.estimate.iGSDBs` were to use LIMMA (which is necessary for unbalanced designs, but not necessary and not used in the current case), then it has to know what the references are in each case; hence the `references <- c("wt", "td")` assignment. It also has be able to figure out which "experimental" (i.e., non-reference) sample goes with which reference. Therefore, they all have to start with the name of their respective reference sample (see the documentation). Having said that, we will ignore the `references` argument for now, and estimate the iGSDBs using averaging, simply as:

```
> iGSDBs.estimated <- dyebias.estimate.iGSDBs(data.norm.est, verbose = TRUE)
```

```
[1] "Estimating the intrinsic gene-specific dyebiases ..."
[1] "Found 4 slides containing 41151 spots"
[1] "Balanced design, simply averaging"
```

```
> summary(iGSDBs.estimated)
```

```
  reporterId           dyebias                A          p.value
 Length:41151       Min.   :-0.715138   Min.   : 8.20   Mode:logical
```

```
 Class :character   1st Qu.:-0.131565   1st Qu.:11.25   NA's:41151
 Mode  :character   Median :-0.019659   Median :11.74
                    Mean   : 0.001517   Mean   :11.70
                    3rd Qu.: 0.111595   3rd Qu.:12.20
                    Max.   : 1.928667   Max.   :14.63
```

5.3 Reading the rest of the data

We now have an estimate of the intrinsic gene specific dye biases, and we can use it to correct all the hybridizations done on platform `GPL5991`. However, that is not yet converted to `marray` objects, so do that first, and also make sure `A` is set and the dye swaps are swapped back:

```
> subset <- sapply(GSMList(gse), function(x) {
+     Meta(x)$platform_id
+ }) == "GPL5991"
> slide.ids.all <- sapply(GSMList(gse), function(x) {
+     Meta(x)$geo_accession
+ })[subset]
> data.raw.all <- dyebias.geo2marray(gse = gse, slide.ids = slide.ids.all,
+     type = "raw", gene.selector = gene.selector, reporterid.name = reporterid.name,
+     cy3.name = cy3.name, cy5.name = cy5.name, Rf.name = Rf.name,
+     Gf.name = Gf.name, Rb.name = Rb.name, Gb.name = Gb.name,
+     )
> data.norm.all <- dyebias.geo2marray(gse = gse, slide.ids = slide.ids.all,
+     type = "norm", gene.selector = gene.selector, reporterid.name = reporterid.name,
+     cy3.name = cy3.name, cy5.name = cy5.name, M.name = M.name)
> maA(data.norm.all) <- log2(maRf(data.raw.all) * maGf(data.raw.all))/2
> swap = c(1, 2, 3, 5, 7, 8, 9)
> maM(data.norm.all)[, swap] <- -maM(data.norm.all)[, swap]
```

5.4 Applying the dye bias correction

As before, we will exclude some spots, based on their intensity, from dye bias correction. The data set does have its own backgrounds, but we know little about the suitability of the spots to act as estimators of the slide bias, so well set estimator.subset=T. This means that no spot is excluded from being an estimator, apart from the second condition, which is that they have an average signal `A` falling within the interquartile range (see the documentation of `dyebias.apply.correction`).

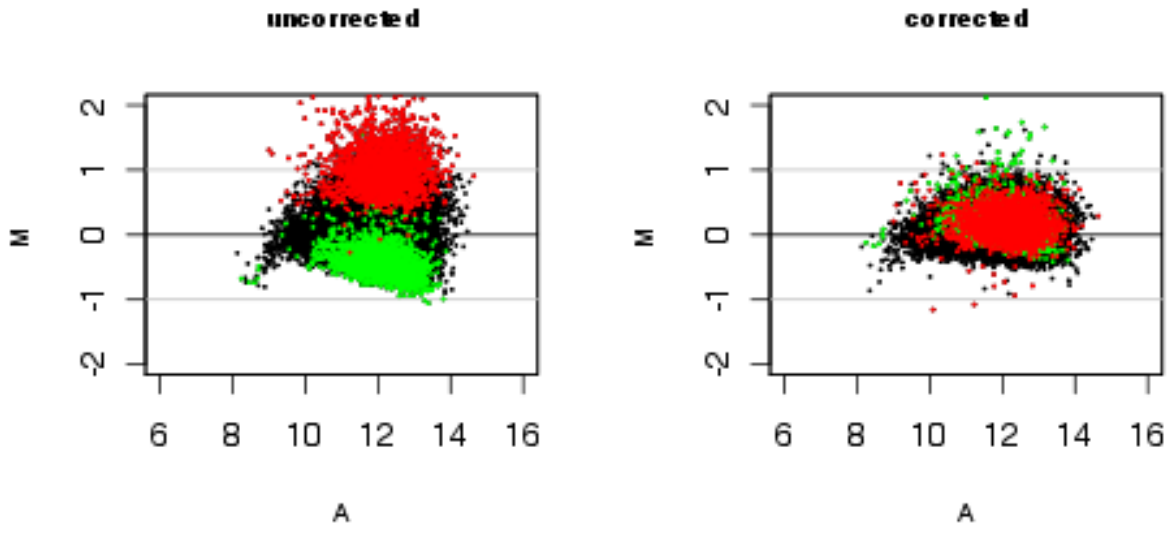
```
> application.subset <- ((maW(data.norm.all) == 1) & dyebias.application.subset(data.raw = data
+     min.SNR = 1.5, maxA = 15, use.background = TRUE))
> correction <- dyebias.apply.correction(data.norm = data.norm.all,
+     iGSDBs = iGSDBs.estimated, estimator.subset = T, application.subset = application.subset,
+     verbose = FALSE)
> correction$summary[, c("slide", "avg.correction", "var.ratio",
+     "reduction.perc", "p.value")]
```

```
  slide avg.correction var.ratio reduction.perc       p.value
1     1     0.51215510 0.9379846      6.2015355  8.874239e-11
2     2     0.31786244 0.9608192      3.9180757  5.086446e-05
3     3     0.40734879 0.9442525      5.5747538  6.066169e-09
4     4     1.51651646 0.7528297     24.7170293 1.109364e-181
5     5     0.73642382 0.8506435     14.9356539  3.113881e-60
6     6     1.73859250 0.2636641     73.6335888  0.000000e+00
7     7     0.02043533 1.0035940     -0.3594009  7.160967e-01
8     8    -0.51330577 0.9988518      0.1148233  9.077435e-01
9     9    -0.69424288 0.8383672     16.1632787  8.187852e-71
```

5.5 Plotting the data

Some people prefer MA-plots over RG-plots, so let's see how the dyebias correction turned out for slide 6, which had the greatest bias:

```
> layout(matrix(1:2, nrow = 1, ncol = 2))
> dyebias.maplot(data = data.norm.all, slide = 6, iGSDBs = iGSDBs.estimated,
+     application.subset = application.subset, main = "uncorrected",
+     cex = 0.2, cex.lab = 0.8, cex.main = 0.8, output = NULL)
> dyebias.maplot(data = correction$data.corrected, slide = 6, iGSDBs = iGSDBs.estimated,
+     application.subset = application.subset, main = "corrected",
+     cex = 0.2, cex.lab = 0.8, cex.main = 0.8, output = NULL)
```

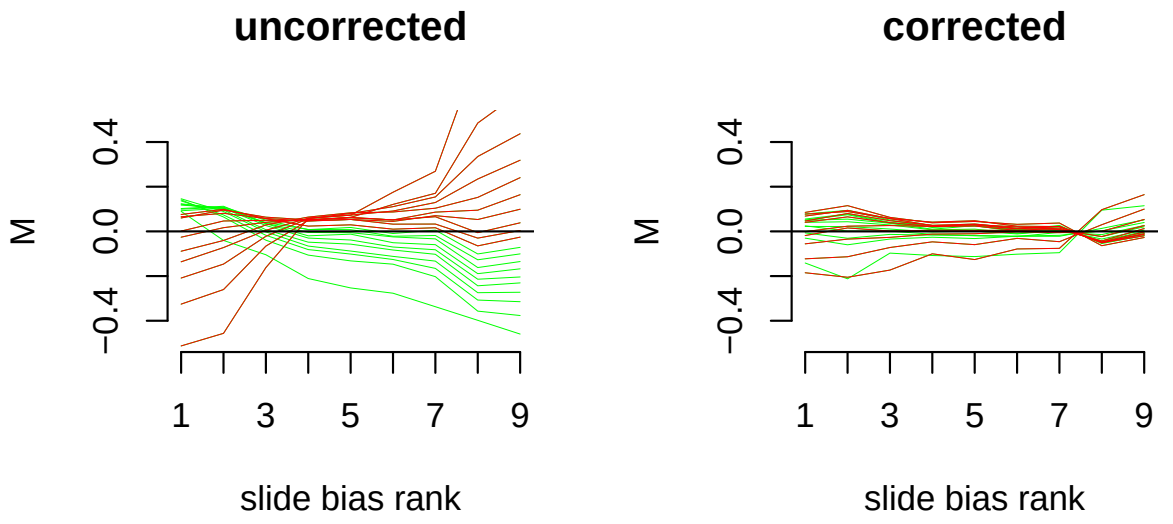


The difference between the uncorrected and corrected data is quite big. The strongly red-biased spots had an average two-fold change, which, after correction, turns out to have been spurious. A number of strongly green-biased spots appear depleted before correction, but may actually have been enriched in the immunoprecipitated samples after correction.

What is also visible is that the data is not properly normalized. The left panel shows a cloud whose bottom has a the downward-sloping trend frequently found in ChIP-on-chip data. This should not be present in normalized data[3]. The dye bias correction procedure tries to also correct this ChIP-on-chip artifact, resulting in suboptimal behaviour (see below).

To get a graphical overview of all corrections, `dyebias.boxplot` could be used, but another function is provided that can show to what extent the data supports our claim that the total dye bias is the product of the slide bias and the intrinsic bias. We order all slides by the slide bias, and plot each spot. However, the number of spots is too large to judge whether their behaviour is linear. The function `dyebias.trendplot` therefore bins all probes by their intrinsic bias, and plots the median of each bin for each slide:

```
> layout(matrix(1:2, nrow = 1, ncol = 2))
> order = dyebias.trendplot(data = data.norm.all, iGSDBs = iGSDBs.estimated,
+     application.subset = application.subset, order = NULL, lwd = 0.1,
+     ylim = c(-0.5, 0.5), output = NULL, main = "uncorrected")
> order = dyebias.trendplot(data = correction$data.corrected, iGSDBs = iGSDBs.estimated,
+     application.subset = application.subset, order = order, lwd = 0.1,
+     ylim = c(-0.5, 0.5), output = NULL, main = "corrected")
```



The method does a good job, although usually the results are even better, with all lines hovering around $M = 0$ after the correction. We attribute this behaviour to the fact that the data is poorly normalized, as mentioned before. This affects the iGSDB estimates, which in turn impacts the correction of the rest of the data. The overall variance, however, is clearly reduced.

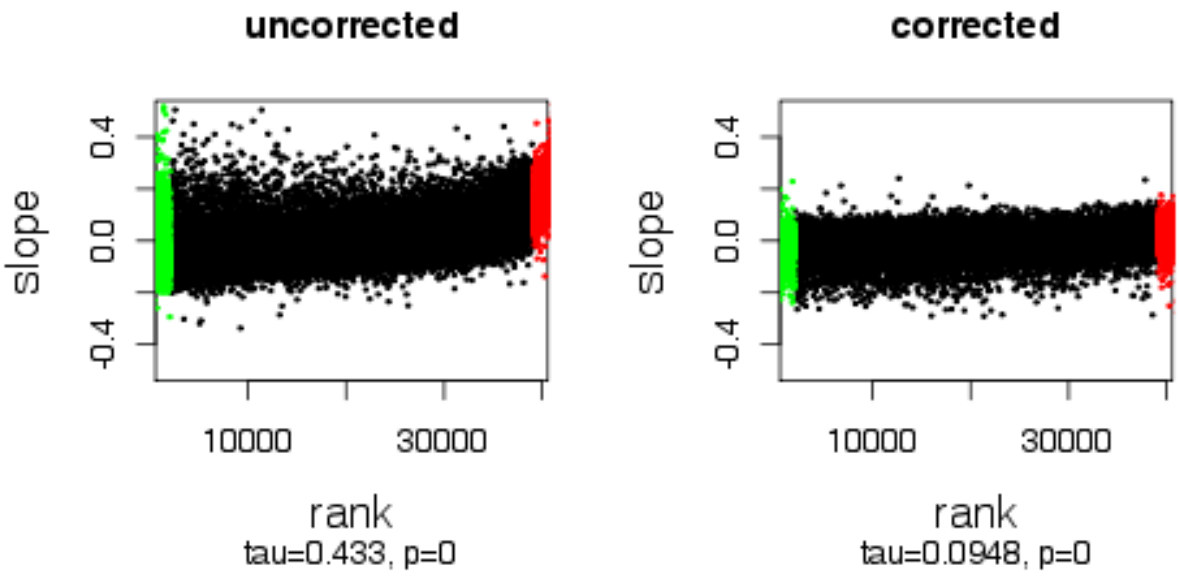
From the left panel it is also clear that the total dye bias shows a consistent trend in each of the bins. This can only happen if the total dye bias is a monotonic function of the slide bias and

[3]We did not renormalise this data as the point of this document is to demonstrate dye bias correction of existing, normalized data.

the intrinsic gene-specific dye bias. Our proposal to simply use the product of both appears to be a good first approximation.

This can also be seen using the `dyebias.monotonicityplot`. It orders all slides by their slide bias, then calculates linear regression lines for the apparent M of each gene. If the monotonicity assumption is warranted, the slopes of each of the regression lines should increase with increasing iGSDB. This can be tested using the Mann-Kendall test, which is available from the `dyebias.monotonicity` function. The `dyebias.monotonicityplot` function plots, for each gene, the slope of each regression line in the data set. This function takes very long to compute, and we do in general not recommend its use for each set of hybridizations that is being corrected. It is included here, however, to further support our model of gene specific dye bias.

```
> layout(matrix(1:2, nrow = 1, ncol = 2))
> order = dyebias.monotonicityplot(data = data.norm.all, iGSDBs = iGSDBs.estimated,
+     order = NULL, ylim = c(-0.5, 0.5), output = NULL, main = "uncorrected")
> order = dyebias.monotonicityplot(data = correction$data.corrected,
+     iGSDBs = iGSDBs.estimated, order = order, ylim = c(-0.5,
+         0.5), output = NULL, main = "corrected")
```



Before dye bias correction with GASSCO, the data shows a clear trend of increasing apparent M, due to dye bias. After correction, this trend has been strongly reduced, supporting our assumption of monotonicity in this case.

6 Closing remarks

So far, we have not found a data set that does not benefit from GASSCO. The code has been in continuous use in our laboratory since early 2008, totalling hundreds of hybrizations. In our

experience, the code and the estimates are very robust, which we attribute to the use of medians and large samples wherever appropriate.

The quality of dye bias correction with GASSCO depends on the accuracy of the iGSDB estimate. This, in turn, depends on having a representative data set of self-self and/or dye-swapped hybrizations. In our experience, a set of around 12 hybrizations is optimal. However, even as few as four hybrizations can result in good dye bias correction, as witnessed by the examples given here, and also by the `data.norm` data set used throughout the `example()` code of the `dyebias` package.

Make sure the data is properly normalized, and that the dye orientation is the same as in the original hybridizations (that is, do not swap them back).

Lastly, we appreciate all feedback.

References

T. Barrett, D. Troup, S. Wilhite, P. Ledoux, D. Rudnev, C. Evangelista, I. Kim, A. Soboleva, M. Tomashevsky, and R. Edga. NCBI GEO: mining tens of millions of expression profiles–database and tools update. *Nucleic Acids Research*, 35:D760–D765, 2007. doi: 10.1093/nar/gkl887.

S. Chen, M. de Vries, and S. Bell. Orc6 is required for dynamic recruitment of Cdt1 during repeated Mcm2-7 loading. *Genes Dev.*, 21:2897–2907, 2007. doi: 10.1101/gad.1596807.

S. Davis and P. Meltzer. GEOquery: a bridge between the Gene Expression Omnibus (GEO) and BioConductor. *Bioinformatics*, 23:1846–1847, 2007. doi: 10.1093/bioinformatics/btm254.

T. Margaritis, P. Lijnzaad, D. van Leenen, D. Bouwmeester, P. Kemmeren, S. van Hooff, and F. Holstege. Adaptable gene-specific dye bias correction for two-channel DNA microarrays. *Molecular Systems Biology*, 5:266, 2009. doi: 10.1038/msb.2009.21.

H. Parkinson, M. Kapushesky, M. Shojatalab, N. Abeygunawardena, R. Coulson, A. Farne, E. Holloway, N. Kolesnykov, P. Lilja, M. Lukk, R. Mani, T. Rayner, A. Sharma, E. William, U. Sarkans, and A. Brazma. ArrayExpress – public database of microarray experiments and gene expression profiles. *Nucleic Acids Research*, 35:D747–D750, 2007. doi: 10.1093/nar/gkl995.

G. K. Smyth. *Limma: linear models for microarray data*, pages 397–420. Springer, New York, 2005.

G. Tuteja, S. Jensen, P. White, and K. KH. Cis-regulatory modules in the mammalian liver: composition depends on strength of Foxa2 consensus site. *Nucleic Acids Research*, 36:4149–4157, 2008. doi: 10.1093/nar/gkn366.

Y. H. Yang, A. Paquet, and S. Dudoit. *marray: Exploratory analysis for two-color spotted microarray data*, 2007. URL `http://www.maths.usyd.edu.au/u/jeany/`. R package version 1.20.0.

7 sessionInfo

- R version 2.11.1 (2010-05-31), `i486-pc-linux-gnu`

- Locale: `C`
- Base packages: base, datasets, grDevices, graphics, methods, stats, utils
- Other packages: Biobase 2.6.1, GEOquery 2.12.0, RCurl 1.3-1, bitops 1.0-4.1, convert 1.24.0, dyebias 1.6.0, dyebiasexamples 1.0.6, limma 3.4.0, marray 1.26.0

This document was generated using Thibaut Jombart's modified `Sweave` function in order to get bitmaps, and a slightly modified version of the original `dyebias/inst/doc/dyebias.Rnw` in order to get the full results (which take too long to compute).