# Normalisation with the normalizeByReference function in the tilingArray package

Wolfgang Huber

April 27, 2009

## Contents



## 1 Introduction

The purpose of this vignette is to demonstrate the probe-response normalization provided by the function `normalizeByReference` in the *tilingArray* package, and to assess its performance. We use the example data from our paper [1]; the data are provided in the *davidTiling* package. Note that some of the computations in this vignette will take several minutes and require Gigabytes of RAM.

```
> library("tilingArray")
> library("Scerevisiaetilingprobe")
> library("davidTiling")
> library("vsn")
> library("RColorBrewer")
> library("Biostrings")

> data("davidTiling")
> class(davidTiling)

[1] "ExpressionSet"
attr(,"package")
[1] "Biobase"
```

```
> dim(exprs(davidTiling))

[1] 6553600       8

> sampleNames(davidTiling)

[1] "09_11_04_S96_genDNA_16hrs_45C_noDMSO.cel"
[2] "041119_S96genDNA_re-hybe.cel"
[3] "041120_S96genDNA_re-hybe.cel"
[4] "05_04_27_2xpolyA_NAP3.cel"
[5] "05_04_26_2xpolyA_NAP2.cel"
[6] "05_04_20_2xpolyA_NAP_2to1.cel"
[7] "050409_totcDNA_14ug_no52.cel"
[8] "030505_totcDNA_15ug_affy.cel"
```

`davidTiling` contains 8 arrays with 6553600 features each. Three of them were hybridized to genomic DNA, which will use as a reference for the normalization, and five to RNA.

## 2 Selection of perfect match and background features

The design of the chip includes about 3 Mio *perfect match probes*, about 3 Mio *mismatch probes*, and a few thousand controls. The `probeAnno` object contains annotations for each probe. It was obtained by aligning the probe sequence to the genomic sequence of S. cerevisiae in August 2005. Please see its manual page and the script `makeProbeAnno.R` in the *davidTiling* package for details.

```
> data("probeAnno")
```

We define two vectors: `whPM` contains indices of probes that have a unique perfect match anywhere in the genome, `whBG` contains a subset of `whPM` with probes whose match is outside any annotated feature on either strand. In the package *davidTiling*, we have already provided two functions `PMindex` and `BGindex` that extract these index vectors from the `probeAnno` environment. If you would like to transfer this analysis to another array type, you will need to compute these sets yourself, essentially by aligning the microarray probes to the target genome and transcriptome.

```
> whPM = PMindex(probeAnno)
> whBG = BGindex(probeAnno)

> length(whPM)

[1] 3049103

> length(whBG)

[1] 829938

> all(whBG %in% whPM)

[1] TRUE
```

There are $6553600 - 3049103 = 3504497$ probes that are not in `whPM`. We will not try to normalize their signal, since we only have meaningful reference intensities for the perfect match probes. For the background estimation, we will use the 829938 features for which we expect no specific signal. Some of them will indeed have specific signal, due to new transcripts that were not yet annotated in SGD in August 2005; but this is not a concern, since we will only use statistical properties of the signal distribution that are insensitive to a small set of outliers. Alternatively, as we show below, we could also use the signal from the MM probes for background estimation.

# 3 Normalization

We select the 5 arrays that had RNA hybridized to them and the 3 arrays that had DNA hybridized to them. We want to normalize the intensity readings from the former by using the latter as a reference.

```
> isRNA = davidTiling$nucleicAcid %in% c("poly(A) RNA","total RNA")
> isDNA = davidTiling$nucleicAcid %in% "genomic DNA"
> stopifnot(sum(isRNA)==5, sum(isDNA)==3)

> pfn = sprintf("assessNorm-normalize%d.pdf", seq(along=which(isRNA)))
> xn2 = normalizeByReference(davidTiling[,isRNA], davidTiling[,isDNA],
+   pm=whPM, background=whBG, plotFileNames=pfn)
```

# 4 Alternative normalization methods

## 4.1 Without dropping the worst 5% probes

For comparison, we also compare to the situation in which we do not throw out the weakest features, by setting `cutoffQuantile=0`.

```
> xn1 = normalizeByReference(davidTiling[,isRNA], davidTiling[,isDNA],
+   pm=whPM, background=whBG, cutoffQuantile=0)
```

## 4.2 Background correction by MM

Instead of the background correction based on "similar" no-target probes that is done in `normalizeByReference`, we can also consider background correction by MM probes.

There is a slight complication: the set of probes that we are considering as PM in this document is somewhat different from those that went into the design of PM/MM pairs on the array, since different versions of the yeast genome sequence were used. We take the intersection. For this, we determine the indices of the designed PM/MM pairs. The array has 2560 rows and 2560 columns. If we count the rows and columns from 0 to 2559, then the linear indices of the features (e. g. in the expression matrix of the *eSet* `davidTiling`) are given by r*2560+c. The PM features lie in rows $1, 3, \ldots, 2557$, their corresponding MM features in rows $2, 4, \ldots, 2558$. We are going to use this information, as well as the probe sequences that are provided in the *Scerevisiaetilingprobe* package.

```
> nc = as.integer(2560)
> stopifnot(nc*nc==nrow(davidTiling), nc*nc==length(Scerevisiaetilingprobe$sequence))
> ipm = rep(as.integer(seq(1, nc-3, by=2)), each=nc) * nc + seq_len(nc)
> seqPM = Scerevisiaetilingprobe$sequence[ipm]
> seqMM = Scerevisiaetilingprobe$sequence[ipm+nc]
> haveSeq = which(! (is.na(seqPM) | is.na(seqMM)) )
> seqPM = DNAStringSet( seqPM[haveSeq] )  ## DNAStringSet does not like NAs
> seqMM = DNAStringSet( seqMM[haveSeq] )
> ipm   = ipm[haveSeq]
> parts = threebands(seqPM, start=13, end=13)
> seqPMcomp = xscat(parts$left, complement(parts$middle), parts$right)
> ispair =  (as(seqMM, "character") == as(seqPMcomp, "character" ))
```

We see that out of the 3270435 probes in odd-numbered rows, listed by `ipm`, most do indeed have a corresponding mismatch probe in the row below.

```
> table(ispair)
```

```
ispair
  FALSE    TRUE
 125913 3144522
```

We define vectors `PMind` and `MMind` that contain the indices of PM/MM pairs,

```
> PMind = ipm[ispair]
> MMind = PMind + nc
```

and a function `normalizeByReferenceWithMM` that is similar to `normalizeByReference` in the *tilingArray* package, except for that for each PM probe it uses the corresponding MM value for the background correction rather than the background estimator that is used in `normalizeByReference`.

```
> normalizeByReferenceWithMM = function(x, reference, pm, mm, cutoffQuantile=0.05) {
+
+   refSig = 2^rowMeans(log2(exprs(reference)[pm,,drop=FALSE]))
+
+   xn = (exprs(x)[pm,] - exprs(x)[mm,]) / refSig
+   yn = exprs(vsn2(xn, lts.quantile=0.95, subsample=20000L, verbose=FALSE))
+
+   throwOut = (refSig < quantile(refSig, probs=cutoffQuantile))
+   yn[throwOut, ] = NA
+
+   res = matrix(as.numeric(NA), nrow=nrow(x), ncol=ncol(yn))
+   res[pm, ] = yn
+
+   return(res)
+ }
> xwmm = normalizeByReferenceWithMM(davidTiling[,isRNA], davidTiling[,isDNA],
+               PMind, MMind)
```

# 5 Assessment

## 5.1 Visually

We would like to visualize the data along genomic coordinates. We select the features that map to the "-" strand of chromosome 9. The integer vectors `sta` and `end` contain the start and end coordinate of their match, `ind` their indices in the array `exprs(davidTiling)`.

```
> sta = probeAnno$"9.-.start"
> end = probeAnno$"9.-.end"
> ind = probeAnno$"9.-.index"
```

We construct a list of vectors, each containing different versions of the intensity data, in order that corresponds to `sta` and `ind` from above.

```
> dat = vector(mode="list", length=5)
> dat[[1]] = log2(exprs(davidTiling)[ind, which(isDNA)[1]])
> dat[[2]] = log2(exprs(davidTiling)[ind, which(isRNA)[1]])
> dat[[3]] = dat[[2]]-dat[[1]]
> dat[[4]] = exprs(xn1)[ind, 1]
> dat[[5]] = exprs(xn2)[ind, 1]
> dat[[6]] = xwmm[ind, 1]
```

```
> for(j in 3:length(dat))
+   dat[[j]] = dat[[j]] - quantile(dat[[j]], 0.05, na.rm=TRUE)
> names(dat) = letters[seq(along=dat)]
```

We select a 10kB region around the highly expressed genes RPN2 and SER33 to fit on a plot, and set the $y$-axis limits:

```
> sel = (sta>=216600 & end<=227000)
> ysc = sapply(dat, function(py) quantile(py, probs=c(0, 1), na.rm=TRUE))
> ysc[, 3:6] = c(-3,8)
```

Now we are ready to plot:

```
> anno = data.frame(start=c(217860, 221078),
+                   end  =c(220697, 222487),
+                   name =I(c("RPN2", "SER33")))
> ticks = c(217, 223, 224, 225, 226)
> comparisonPlot((sta+end)[sel]/2, lapply(dat, "[", sel), yscale=ysc,
+   anno=anno, ticks=ticks, cex=0.2)
```

The result is shown in Figure 1. It shows scatterplots of different types of signal ($y$-axis) along genomic coordinates ($x$-axis). Each dot corresponds to a microarray feature. Note how the signal distributions in panels b)-f) vary both with respect to the variability within transcribed and untranscribed segments, and in the amplitude between them. a) signal from one of the DNA hybridizations (logarithmic scale, base 2). The $y$-coordinate of each dot is also encoded using a pseudo-color scheme. Dark red corresponds to features that have a very weak response, dark blue to those with the strongest response. The same coloring is also used in panels b)-f). b) unnormalized intensities from one of the poly(A) RNA bybridizations (logarithmic scale, base 2). c) Divide RNA-signal by DNA-signal then take logarithm (base 2). d) Background subtraction of the RNA-signal, divide by DNA-signal, then variance stabilizing normalization (vsn, glog base 2). e) In addition to d), drop the 5% weakest features in the DNA hybridization. f) Similar to e), but using the MM probes for background correction instead.

Now here's a bit of a hack: the plot symbols are too big if the plot is produced as above, and somehow the PDF and EPS drivers of R ignore the `cex` parameter. However, one can open the EPS file and replace all occurences of the string "3.00" by "1.00".

```
> writeLines(sub(" 3.00", " 1.50",
+     readLines("assessNorm-alongChromPlot.eps"), fixed=TRUE),
+  con=("assessNorm-tmp.eps"))
> system("ps2pdf assessNorm-tmp.eps assessNorm-alongChromPlot.pdf")
```

## 5.2 Quantitatively

In order to assess to quantitatively assess the results of normalization, we consider a signal/noise ratio. Note that only looking at one or another (signal, or noise) by itself could be misleading.

We define a set of control regions, which correspond to either highly expressed transcripts (`positiveCtrls`) or untranscribed intergenic regions (`negativeCtrls`). The assumption is that the signal within a region should be constant, and deviations from that are "noise", while the difference between positve and negative controls should be large, and is counted as "signal".

```
> positiveCtrls = cbind(c(217860,220697), ## RPN2
+                       c(221078,222487)) ## SER33         .
> negativeCtrls = cbind(c(216800, 217700),
+                       c(222800, 227000)) ## SPO22
```

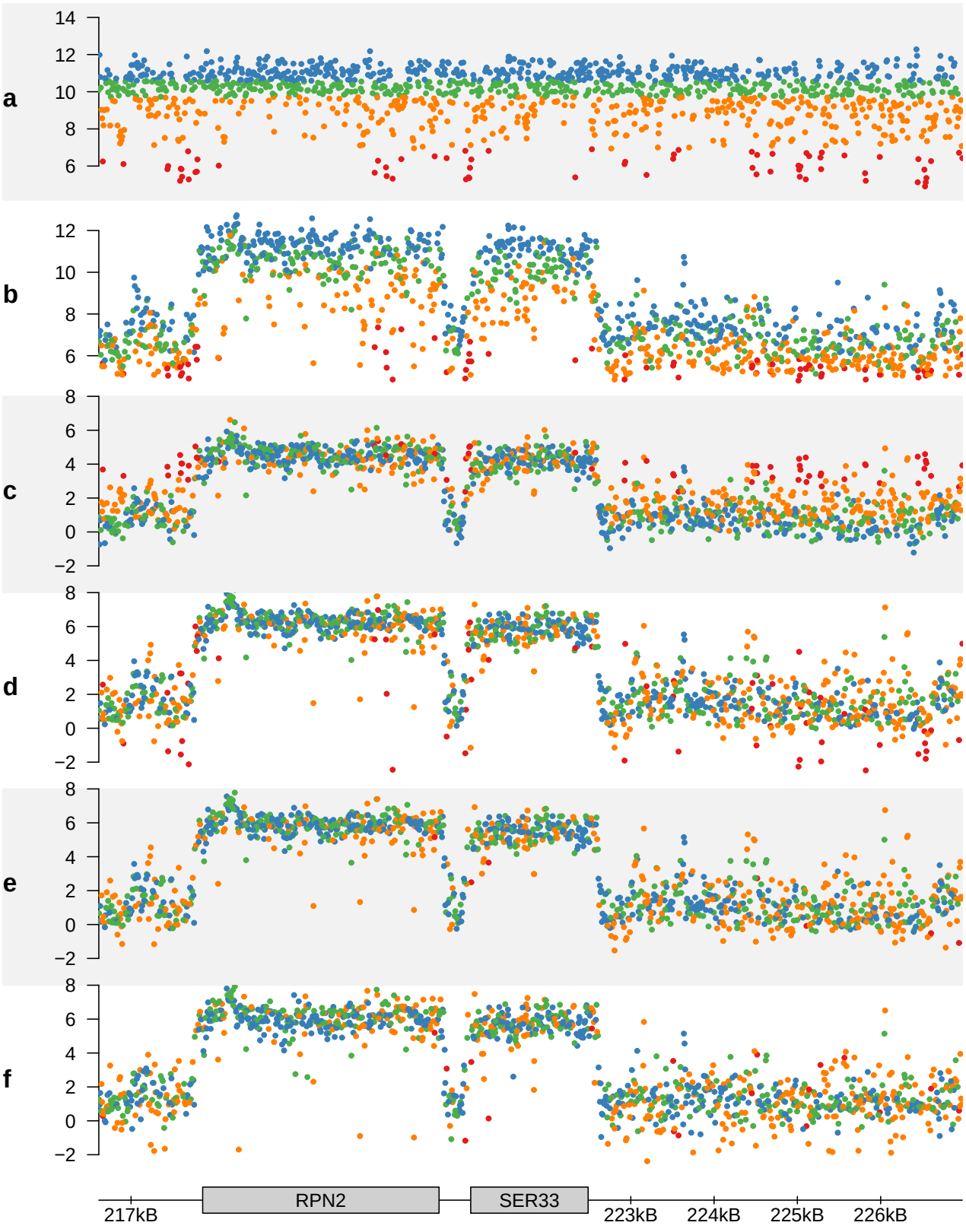


Figure 1: Along-chromosome plots of the data for different normalization methods. Please see text.

Noise $\sigma$ is calculated as the average of the differences between 97.5% and 2.5% quantiles of the data within each of the control regions. The range between the 97.5% and 2.5% quantiles contains 95% of the data, while 5% is outside the range.

$$\sigma = \frac{1}{Q_N^{0.975} - Q_N^{0.025}} \cdot \frac{\sum_{r \in \{\text{pos,neg}\}} Q_r^{0.975} - Q_r^{0.025}}{|\{\text{pos, neg}\}|}. \tag{1}$$

Here, the symbol $r$ counts over the different regions. The constant in the denominator is the differences between 95% and 5% quantiles for the standard Normal distribution, hence $\sigma$ is equal to 1 if the data come from the standard Normal distribution.

Signal $\Delta\mu$ is calculated as the difference between the averages of the means of positive and negative control regions.

$$\Delta\mu = \frac{\sum_{r \in \{\text{pos}\}} \mu_r}{|\{\text{pos}\}|} - \frac{\sum_{r \in \{\text{neg}\}} \mu_r}{|\{\text{neg}\}|}. \tag{2}$$

I have explored many variations of this calculation, using different definitions of $\sigma$, $\Delta\mu$, and of the control regions. The ranking (relative order) of the methods was always the same as shown in the following.

```
> fac = 2*qnorm(0.975)
> withinAndBetween = function(x, ...) {
+   meanAndSd = function(region, dohist=FALSE) {
+     d = x[(sta>=region[1]) & (end<=region[2]) & !is.na(x)]
+     res = c(mean(d), diff(quantile(d, c(0.025, 0.975)))/fac, length(d))
+     if(dohist) hist(d, 20, main=paste(signif(res, 3)), col="orange")
+     res
+   }
+   p = apply(positiveCtrls, 2, meanAndSd, ...)
+   n = apply(negativeCtrls, 2, meanAndSd, ...)
+   dmu =  sum(p[1,]*p[3,])/sum(p[3,]) - sum(n[1,]*n[3,])/sum(n[3,])
+   sig = (sum(p[2,]*p[3,])+sum(n[2,]*n[3,])) / (sum(p[3,])+sum(n[3,]))
+   return(c("S/N"=dmu/sig, "S"=dmu, "N"=sig))
+ }

> sn = sapply(dat[2:6], withinAndBetween, dohist=TRUE)

       b     c    d     e    f
S/N 3.22 3.473 4.13 4.653 4.40
S   3.67 3.109 4.48 4.461 4.60
N   1.14 0.895 1.09 0.959 1.05
```

### 5.3 Dependency of the signal on GC content before and after normalization

We calculate the GC content of each probe,

```
> bc = alphabetFrequency(seqPM, baseOnly=TRUE)
> nrGC = bc[,"C"]+bc[,"G"]
> rggc = c(4,18)
> sel = (nrGC>=rggc[1] & nrGC<=rggc[2])
> nrGC = nrGC[sel]
> myPlot = function(x, tit) {
+   maxval = quantile(x, probs=0.9999, na.rm=TRUE)
```

```
+   minval = maxval-10.5
+   mycol = colorRampPalette(brewer.pal(9, "GnBu"))(diff(rggc)+1)
+   boxplot(x[sel]~nrGC, ylim=c(minval,maxval), main="", col=mycol, outline=FALSE)
+   text(rggc[1]-5, maxval+1, tit, pos=2, font=2, xpd=NA)
+ }
```

Remember that we defined `seqPM` in Section 4.2.

```
> par(mfrow=c(1,2))
> dat1 = log2(exprs(davidTiling)[ipm, which(isRNA)[1]])
> dat2 = exprs(xn2)[ipm, 1] - quantile(exprs(xn2)[ipm, 1], 0.05, na.rm=TRUE)
> myPlot(dat1, "a")
> myPlot(dat2, "b")
```

The result is shown in Figure 2.

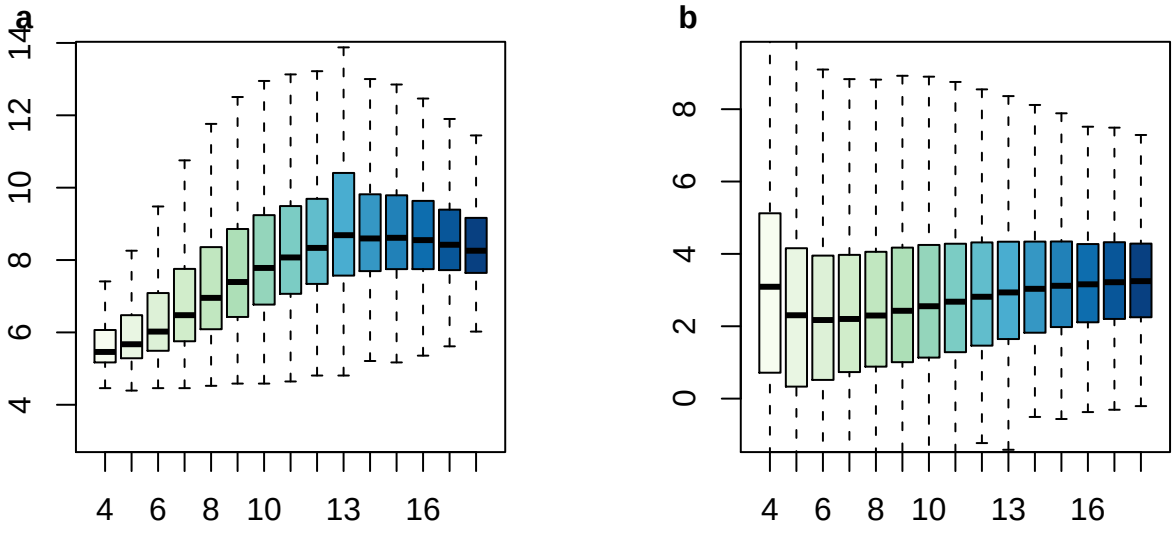


Figure 2: ECDF of $\log_2$ intensities (a) and normalized intensities (b) stratified by GC content.

## 6 Version Information

This vignette was generated using the following package versions:

```
> toLatex(sessionInfo())
```

- R version 2.10.0 Under development (unstable) (2009-04-12 r48319), `x86_64-unknown-linux-gnu`
- Locale: `LC_CTYPE=C;LC_NUMERIC=C;LC_TIME=C;LC_COLLATE=C;LC_MONETARY=C;LC_MESSAGES=it_IT;LC_PAPE`

- Base packages: base, datasets, grDevices, graphics, methods, stats, utils
- Other packages: AnnotationDbi 1.5.23, Biobase 2.3.11, Biostrings 2.11.56, DBI 0.2-4, GO.db 2.2.11, IRanges 1.1.69, RColorBrewer 1.0-2, RSQLite 0.7-1, Scerevisiaetilingprobe 1.2.0, davidTiling 1.2.8, fortunes 1.3-6, pixmap 0.4-9, tilingArray 1.21.12, vsn 3.10.8
- Loaded via a namespace (and not attached): KernSmooth 2.22-22, affy 1.21.10, affyio 1.11.3, annotate 1.21.5, genefilter 1.23.4, grid 2.10.0, lattice 0.17-22, limma 2.17.21, preprocessCore 1.5.3, splines 2.10.0, strucchange 1.3-7, survival 2.35-4, tools 2.10.0, xtable 1.5-5

# References

[1] Lior David, Wolfgang Huber, Marina Granovskaia, Joern Toedling, Curtis J. Palm, Lee Bofkin, Ted Jones, Ronald W. Davis, and Lars M. Steinmetz A high-resolution map of transcription in the yeast genome. *PNAS*, 2006. 1