missMethyl: Analysing Illumina HumanMethylation450 BeadChip Data

Belinda Phipson and Jovana Maksimovic

Modified: 28 January, 2015. Compiled: April 16, 2015

Contents

1 Introduction

The *missMethyl* package contains functions to analyse methylation data from Illumina's HumanMethylation450 beadchip. These arrays are a cost-effective alternative to whole genome bisulphite sequencing, and as such are widely used to profile DNA methylation. Specifically, missMethyl contains functions to perform SWAN normalisation [9] and differential variability analysis [10]. As our lab's research into specialised analyses of these arrays continues we anticipate that the package will be continuously updated with new functions.

Raw data files are in IDAT format, which can be read into R using the *minfi* package [1]. Statistical analyses are usually performed on M-values, and β values are used for visualisation, both of which can be extracted from *MethylSet* objects, which is a class of object created by *minfi*. For detecting differentially variable CpGs we recommend that the analysis is performed on M-values. All analyses described here are performed at the CpG site level.

2 Reading data into R

We will use the data in the *minfiData* package to demonstrate the functions in *missMethyl*. The example dataset has 6 samples across two slides. The sample information is in the targets file. An essential column in the targets file is the "Basename" column which tells *minfi* where the idat files to be read in are located. The R commands to read in the data are taken from the *minfi* User's Guide. For additional details on how to read the IDAT files into R, as well as information regarding quality control please refer to the *minfi* User's Guide.

```
library(missMethyl)
```

```
## Setting options('download.file.method.GEOquery'='auto')
## Loading required package:  DBI
```

```
library(limma)
library(minfi)
```

```
## Loading required package:  BiocGenerics
## Loading required package:  parallel
##
## Attaching package:  'BiocGenerics'
##
## The following objects are masked from 'package:parallel':
##
##    clusterApply, clusterApplyLB, clusterCall, clusterEvalQ, clusterExport,
##    clusterMap, parApply, parCapply, parLapply, parLapplyLB, parRapply,
##    parSapply, parSapplyLB
##
## The following object is masked from 'package:limma':
##
##    plotMA
##
## The following object is masked from 'package:stats':
##
##    xtabs
##
## The following objects are masked from 'package:base':
##
##    anyDuplicated, append, as.data.frame, as.vector, cbind, colnames, do.call,
##    duplicated, eval, evalq, Filter, Find, get, intersect, is.unsorted, lapply,
##    Map, mapply, match, mget, order, paste, pmax, pmax.int, pmin, pmin.int,
##    Position, rank, rbind, Reduce, rep.int, rownames, sapply, setdiff, sort,
##    table, tapply, union, unique, unlist, unsplit
##
## Loading required package:  Biobase
## Welcome to Bioconductor
##
##    Vignettes contain introductory material; view with 'browseVignettes()'.  To
##    cite Bioconductor, see 'citation("Biobase")', and for packages
##    'citation("pkgname")'.
##
## Loading required package:  lattice
## Loading required package:  GenomicRanges
## Loading required package:  S4Vectors
## Loading required package:  stats4
## Loading required package:  IRanges
```

```
## Loading required package:  GenomeInfoDb
## Loading required package:  Biostrings
## Loading required package:  XVector
## Loading required package:  bumphunter
## Loading required package:  foreach
## Loading required package:  iterators
## Loading required package:  locfit
## locfit 1.5-9.1 2013-03-22
```

```
library(minfiData)
```

```
## Loading required package:  IlluminaHumanMethylation450kmanifest
## Loading required package:  IlluminaHumanMethylation450kanno.ilmn12.hg19
```

```
baseDir <- system.file("extdata", package = "minfiData")
targets <- read.450k.sheet(baseDir)
```

```
## [read.450k.sheet] Found the following CSV files:
## [1] "/Library/Frameworks/R.framework/Versions/3.2/Resources/library/minfiData/extdata/SampleSheet.csv"
```

```
targets[,1:9]
```

```
##   Sample_Name Sample_Well Sample_Plate Sample_Group Pool_ID person age sex status
## 1    GroupA_3          H5           NA       GroupA      NA    id3  83   M normal
## 2    GroupA_2          D5           NA       GroupA      NA    id2  58   F normal
## 3    GroupB_3          C6           NA       GroupB      NA    id3  83   M cancer
## 4    GroupB_1          F7           NA       GroupB      NA    id1  75   F cancer
## 5    GroupA_1          G7           NA       GroupA      NA    id1  75   F normal
## 6    GroupB_2          H7           NA       GroupB      NA    id2  58   F cancer
```

```
targets[,10:12]
```

```
##    Array      Slide
## 1 R02C02 5723646052
## 2 R04C01 5723646052
## 3 R05C02 5723646052
## 4 R04C02 5723646053
## 5 R05C02 5723646053
## 6 R06C02 5723646053
##
## 1 /Library/Frameworks/R.framework/Versions/3.2/Resources/library/minfiData/extdata/5723646052/572364605
## 2 /Library/Frameworks/R.framework/Versions/3.2/Resources/library/minfiData/extdata/5723646052/572364605
## 3 /Library/Frameworks/R.framework/Versions/3.2/Resources/library/minfiData/extdata/5723646052/572364605
## 4 /Library/Frameworks/R.framework/Versions/3.2/Resources/library/minfiData/extdata/5723646053/572364605
## 5 /Library/Frameworks/R.framework/Versions/3.2/Resources/library/minfiData/extdata/5723646053/572364605
## 6 /Library/Frameworks/R.framework/Versions/3.2/Resources/library/minfiData/extdata/5723646053/572364605
```

```
rgSet <- read.450k.exp(targets = targets)
```

The data is now an *RGChannelSet* object and needs to be normalised and converted to a *MethylSet* object.

3 Subset-quantile within array normalization (SWAN)

SWAN (subset-quantile within array normalization) is a within-array normalization method for Illumina 450k BeadChips. Technical differencs have been demonstrated to exist between the Infinium I and Infinium II assays on a single 450K array [2, 3]. Using the SWAN method substantially reduces the technical variability between the assay designs whilst

maintaining the important biological differences. The SWAN method makes the assumption that the number of CpGs within the 50bp probe sequence reflects the underlying biology of the region being interrogated. Hence, the overall distribution of intensities of probes with the same number of CpGs in the probe body should be the same regardless of assay type. The method then uses a subset quantile normalization approach to adjust the intensities of each array [9].

SWAN can take a *MethylSet*, *RGChannelSet* or *MethyLumiSet* as input. It should be noted that, in order to create the normalization subset, SWAN randomly selects Infinium I and II probes that have one, two and three underlying CpGs; as such, we recommend using `set.seed` before `SWAN` to ensure that the normalized intensities will be identical, if the normalization is repeated.

The technical differences between Infinium I and II assay designs can result in aberrant beta value distributions (Figure 1, panel "Raw"). Using SWAN corrects for the technical differences between the Infinium I and II assay designs and produces a smoother overall β value distribution (Figure 1, panel "SWAN").

```
mSet <- preprocessRaw(rgSet)
mSetSw <- SWAN(mSet,verbose=TRUE)
```

```
## [SWAN] Preparing normalization subset
## [SWAN] Normalizing methylated channel
## [SWAN] Normalizing array 1 of 6
## [SWAN] Normalizing array 2 of 6
## [SWAN] Normalizing array 3 of 6
## [SWAN] Normalizing array 4 of 6
## [SWAN] Normalizing array 5 of 6
## [SWAN] Normalizing array 6 of 6
## [SWAN] Normalizing unmethylated channel
## [SWAN] Normalizing array 1 of 6
## [SWAN] Normalizing array 2 of 6
## [SWAN] Normalizing array 3 of 6
## [SWAN] Normalizing array 4 of 6
## [SWAN] Normalizing array 5 of 6
## [SWAN] Normalizing array 6 of 6
```

4 Filter out poor quality probes

Poor quality probes can be filtered out based on the detection p-value. For this example, to retain a CpG for further analysis, we require that the detection p-value is less than 0.01 in all samples.

```
detP <- detectionP(rgSet)
keep <- rowSums(detP < 0.01) == ncol(rgSet)
mSetSw <- mSetSw[keep,]
```

5 Extracting β and M-values

Now that the data has been SWAN normalised we can extract β and M-values from the *MethylSet* object. We prefer to add an offset to the methylated and unmethylated intensities when calculating M-values, hence we extract the methylated and unmethylated channels separately and perform our own calculation. For all subsequent analysis we use a random selection of 20 000 CpGs to reduce computation time.

```
mset_reduced <- mSetSw[sample(1:nrow(mSetSw), 20000),]
meth <- getMeth(mset_reduced)
unmeth <- getUnmeth(mset_reduced)
Mval <- log2((meth + 100)/(unmeth + 100))
```

```
par(mfrow=c(1,2), cex=1.25)
densityByProbeType(mSet[,1], main = "Raw")
densityByProbeType(mSetSw[,1], main = "SWAN")
```

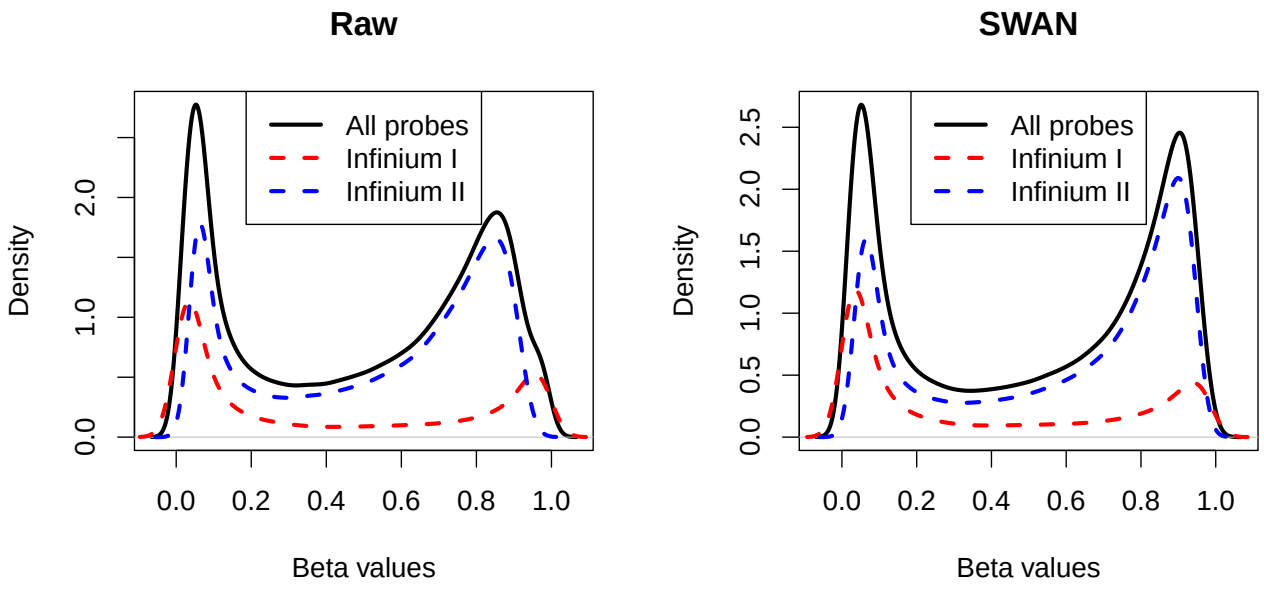

Figure 1: Density distributions of β values before and after using SWAN.

```
beta <- getBeta(mset_reduced)
dim(Mval)
```

```
## [1] 20000     6
```

An MDS plot (Figure 2) is a good sanity check to make sure samples cluster together according to the main factor of interest, in this case, cancer and normal.

6 Testing for differential methylation using limma

To test for differential methylation we use the *limma* package [12], which employs an empirical Bayes framework based on Guassian model theory. First we need to set up the design matrix. There are a number of ways to do this, the most straightforward is directly from the targets file. There are a number of variables, with the status column indicating cancer/normal samples. From the person column of the targets file, we see that the cancer/normal samples are matched, with 3 individuals each contributing both a cancer and normal sample. Since the limma model framework can handle any experimental design which can be summarised by a design matrix, we can take into account the paired nature of the data in the analysis. For more complicated experimental designs, please refer to the *limma* User's Guide.

```
group <- factor(targets$status,levels=c("normal","cancer"))
id <- factor(targets$person)
design <- model.matrix(~id + group)
design
```

```
##   (Intercept) idid2 idid3 groupcancer
## 1           1     0     1           0
## 2           1     1     0           0
```

```
par(mfrow=c(1,1))
plotMDS(Mval, labels=targets$Sample_Name, col=as.integer(factor(targets$status)))
legend("topleft",legend=c("Cancer","Normal"),pch=16,cex=1.2,col=1:2)
```

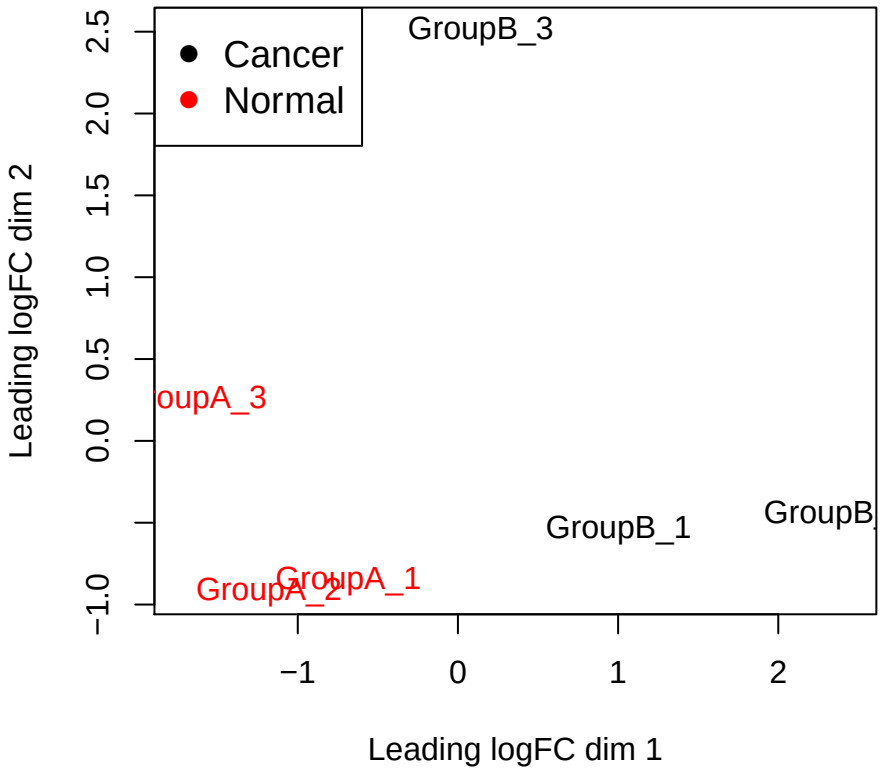

Figure 2: A multi-dimensional scaling (MDS) plot of cancer and normal samples.

```
## 3           1     0     1           1
## 4           1     0     0           1
## 5           1     0     0           0
## 6           1     1     0           1
## attr(,"assign")
## [1] 0 1 1 2
## attr(,"contrasts")
## attr(,"contrasts")$id
## [1] "contr.treatment"
##
## attr(,"contrasts")$group
## [1] "contr.treatment"
```

Now we can test for differential methylation using the `lmFit` and `eBayes` functions from *limma*. As input data we use the matrix of M-values.

```
fit.reduced <- lmFit(Mval,design)
fit.reduced <- eBayes(fit.reduced)
```

The numbers of hyper-methylated (1) and hypo-methylated (-1) can be displayed using the `decideTests` function in *limma* and the top 10 differentially methylated CpGs for cancer versus normal outputted using `topTable`.

```
summary(decideTests(fit.reduced))
```

```
##    (Intercept) idid2 idid3 groupcancer
## -1        6951     0   111         532
## 0         3270 20000 19881       19067
## 1         9779     0     8         401
```

```
top<-topTable(fit.reduced,coef=4)
top
```

```
##               logFC    AveExpr        t      P.Value adj.P.Val        B
## cg25937714 4.220106 -0.2820906 17.28848 9.108985e-06 0.0376256 3.978709
## cg18672939 4.782889 -0.8263457 16.46455 1.167445e-05 0.0376256 3.815880
## cg04832466 4.077194 -1.3088199 15.75744 1.458709e-05 0.0376256 3.663821
## cg13975523 4.167085 -1.0043858 15.51194 1.579561e-05 0.0376256 3.608142
## cg13562911 3.699583 -1.2332523 15.22088 1.738637e-05 0.0376256 3.540092
## cg24385334 3.926623 -1.9771244 15.19175 1.755585e-05 0.0376256 3.533156
## cg06952671 5.251065 -1.7732386 14.56099 2.175708e-05 0.0376256 3.377138
## cg11862642 3.470721 -1.1364987 14.19704 2.472540e-05 0.0376256 3.281784
## cg13631916 4.071110 -0.1024161 13.68208 2.979245e-05 0.0376256 3.139700
## cg17510056 5.154957  0.3324675 13.65250 3.011924e-05 0.0376256 3.131273
```

Note that since we performed our analysis on M-values, the logFC and AveExpr columns are computed on the M-value scale. For interpretability and visualisation we can look at the β values. The beta values for the top 4 differentially methylated CpGs shown in Figure 3.

```
cpgs <- rownames(top)
par(mfrow=c(2,2))
for(i in 1:4){
stripchart(beta[rownames(beta)==cpgs[i],]~design[,4],method="jitter",
group.names=c("Normal","Cancer"),pch=16,cex=1.5,col=c(4,2),ylab="Beta values",
vertical=TRUE,cex.axis=1.5,cex.lab=1.5)
title(cpgs[i],cex.main=1.5)
}
```

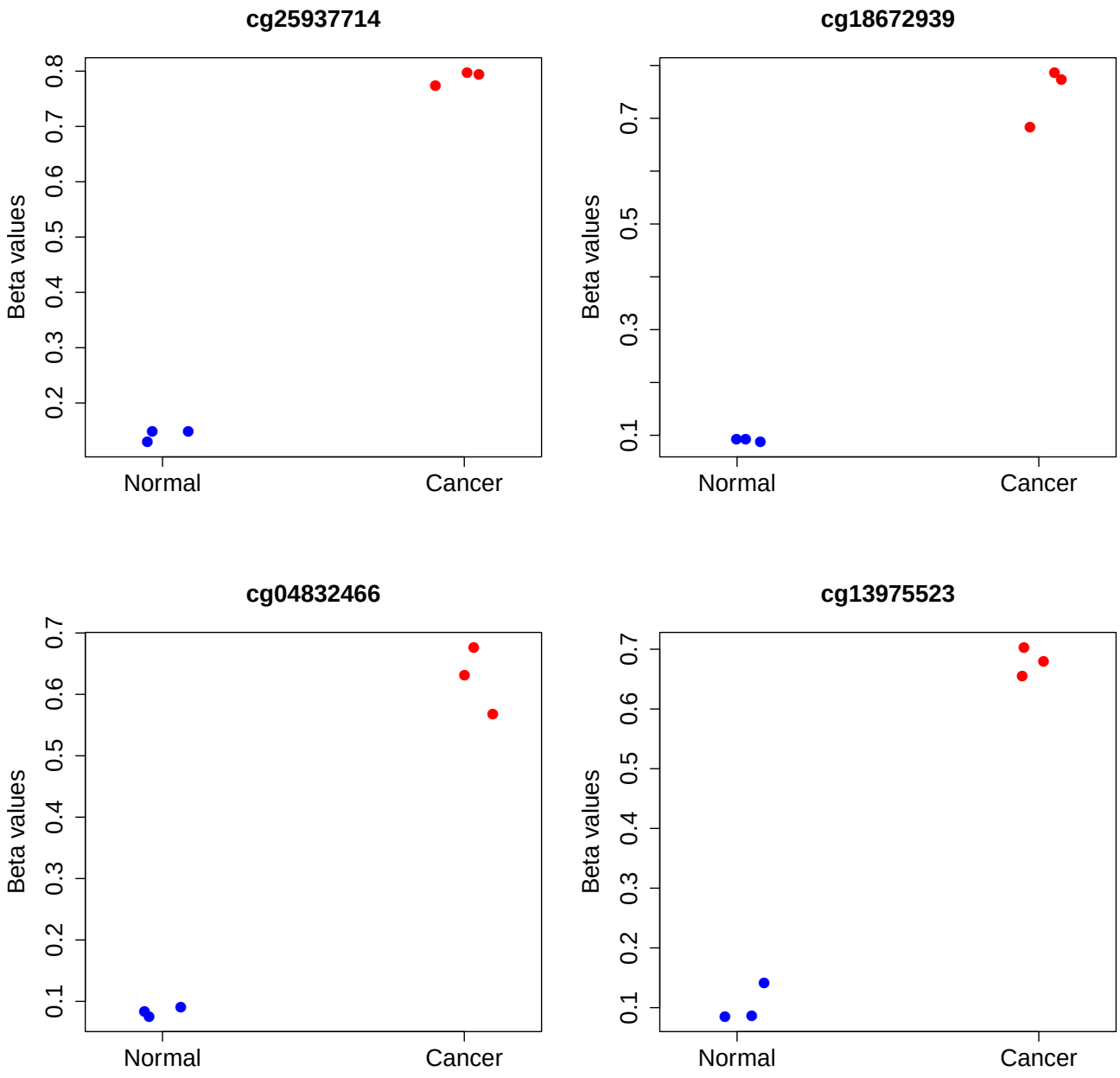

Figure 3: The β values for the top 4 differentially methylated CpGs.

7 Removing unwanted variation when testing for differential methylation

Like other platforms, 450k array studies are subject to unwanted technical variation such as batch effects and other, often unknown, sources of variation. The adverse effects of unwanted variation have been extensively documented in gene expression array studies and have been shown to be able to both reduce power to detect true differences and to increase the number of false discoveries. As such, when it is apparent that data is significantly affected by unwanted variation, it is advisable to perform an adjustment to mitigate its effects.

missMethyl provides a *limma*-like interface to functions from the CRAN *ruv* package that enables the removal of unwanted variation when performing a differential analysis. All of the *ruv* methods rely on negative control features to accurately estimate the components of unwanted variation. Negative control features are probes/genes/etc. that are known *a priori* to not truly be associated with the biological factor of interest, but are affected by unwanted variation. For example, in a microarray gene expression study, these could be house-keeping genes or a set of spike-in controls. Negative control features are extensively discussed in Gagnon-Bartsch and Speed (2012) [4] and Gagnon-Bartsch et al. (2013) [5]. Once the unwanted factors are accurately estimated from the data, they are adjusted for in the linear model that describes the differential analysis.

If the negative control features are not known *a priori*, they can be identified empirically. This can be done by initially performing a differential methylation analysis without any adjustment using *limma*, as described in the previous section. For simplicity, we are ignoring the paired nature of the cancer and normal samples in this example.

If the negative control features are not known *a priori*, they can be identified empirically. This can be achieved using a 2-stage approach, *RUVm*, based on **RUV-inverse**. Stage 1 involves performing a differential methylation analysis with using **RUV-inverse** and the 613 Illumina negative controls (INCs) as negative control features. This will produce a list of CpGs ranked by p-value according to their level of association with the factor of interest. This list can then be used to identify a set of empirical control probes (ECPs), which will capture more of the unwanted variation than using the INCs alone. ECPs are selected by designating a proportion of the CpGs least associated with the factor of interest as negative control features; this can be done based on either an FDR cut-off or by taking a fixed percentage of probes from the bottom of the ranked list. Stage 2 involves performing a second differential methylation ananlysis on the original data using **RUV-inverse** and the ECPs. For simplicity, we are ignoring the paired nature of the cancer and normal samples in this example.

```
# get M-values for ALL probes
meth <- getMeth(mSet)
unmeth <- getUnmeth(mSet)
M <- log2((meth + 100)/(unmeth + 100))

grp <- factor(targets$status,levels=c("normal","cancer"))
des <- model.matrix(~grp)
des
```

```
##   (Intercept) grpcancer
## 1           1         0
## 2           1         0
## 3           1         1
## 4           1         1
## 5           1         0
## 6           1         1
## attr(,"assign")
## [1] 0 1
## attr(,"contrasts")
## attr(,"contrasts")$grp
## [1] "contr.treatment"
```

```
INCs <- getINCs(rgSet)
head(INCs)
```

```
##          5723646052_R02C02 5723646052_R04C01 5723646052_R05C02 5723646053_R04C02
## 13792480        -0.3299654        -1.0955482        -0.5266103        -0.6374299
## 69649505        -1.0354488        -1.4943396        -1.0067050        -0.8854881
## 34772371        -1.1286422        -0.2995603        -0.8192636        -0.6895514
## 28715352        -0.5553373        -0.7599489        -0.7186973        -1.7903619
## 74737439        -1.1169178        -0.8656399        -0.6429681        -0.8872082
## 33730459        -0.7714684        -0.5622424        -0.7724825        -1.5623138
##          5723646053_R05C02 5723646053_R06C02
## 13792480         -1.116598        -0.4332793
## 69649505         -1.586679        -0.9217329
## 34772371         -1.161155        -0.6186795
## 28715352         -1.348105        -1.0067259
## 74737439         -1.064986        -0.9841833
## 33730459         -2.079184        -1.0445246
```

```
Mc <- rbind(M,INCs)
ctl <- rownames(Mc) %in% rownames(INCs)
table(ctl)
```

```
## ctl
##  FALSE   TRUE
## 485512    613
```

```
rfit1 <- RUVfit(data=Mc, design=des, coef=2, ctl=ctl) # Stage 1 analysis
rfit2 <- RUVadj(rfit1)
```

Now that we have performed an initial differential methylation analysis to rank the CpGs with respect to their association with the factor of interest, we can designate the CpGs that are least associated with the factor of interest based on FDR-adjusted p-value as ECPs.

```
top1 <- topRUV(rfit2, num=Inf)
head(top1)
```

```
##            coefficients        t            p      p.BH     p.ebayes p.ebayes.BH
## cg04743961     4.838190 26.74467 3.812882e-05 0.1401969 3.516091e-07  0.01017357
## cg07155336     5.887409 17.62103 1.608653e-04 0.1401969 3.583107e-07  0.01017357
## cg20925841     4.790211 26.69524 3.837354e-05 0.1401969 3.730375e-07  0.01017357
## cg03607359     4.394397 34.74068 1.542013e-05 0.1401969 4.721205e-07  0.01017357
## cg10566121     4.787914 21.80693 7.717708e-05 0.1401969 5.238865e-07  0.01017357
## cg07655636     4.571758 22.99708 6.424216e-05 0.1401969 6.080091e-07  0.01017357
```

```
ctl <- rownames(M) %in% rownames(top1[top1$p.ebayes.BH > 0.5,])
table(ctl)
```

```
## ctl
##  FALSE   TRUE
## 172540 312972
```

We can then use the ECPs to perform a second differential methylation with **RUV-inverse**, which is adjusted for the unwanted variation estimated from the data.

```
# Perform RUV adjustment and fit
rfit1 <- RUVfit(data=M, design=des, coef=2, ctl=ctl) # Stage 2 analysis
rfit2 <- RUVadj(rfit1)

# Look at table of top results
topRUV(rfit2)
```

```
##            coefficients        t            p      p.BH     p.ebayes  p.ebayes.BH
```

```
## cg07155336     5.769286 15.345069 0.002005546 0.3431163 1.434834e-55 6.966293e-50
## cg06463958     5.733093 15.434797 0.001978272 0.3431163 6.749298e-55 1.638433e-49
## cg00024472     5.662959 15.946200 0.001832444 0.3431163 1.319390e-53 2.135266e-48
## cg02040433     5.651399 10.054445 0.005389436 0.3431163 2.146210e-53 2.605027e-48
## cg13355248     5.595396  9.963702 0.005504213 0.3431163 2.234891e-52 2.022589e-47
## cg02467990     5.592707  6.859614 0.013008521 0.3431163 2.499534e-52 2.022589e-47
## cg00817367     5.527501 13.070583 0.002921656 0.3431163 3.710480e-51 2.573547e-46
## cg11396157     5.487992 10.931263 0.004436178 0.3431163 1.873636e-50 1.137091e-45
## cg16306898     5.466780  5.573935 0.020790127 0.3431163 4.448085e-50 2.399554e-45
## cg03735888     5.396242 15.482605 0.001963955 0.3431163 7.700032e-49 3.738458e-44
```

Note, at present `RUVfit` does not support contrasts, so only one factor of interest can be interrogated at a time using a design matrix with an intercept term.

8 Testing for differential variability (DiffVar)

8.1 Methylation data

Rather than testing for differences in mean methylation, we may be interested in testing for differences between group variances. For example, it has been hypothesised that highly variable CpGs in cancer are important for tumour progression [7]. Hence we may be interested in CpG sites that are consistently methylated in the normal samples, but variably methylated in the cancer samples. In general we recommend at least 10 samples in each group for accurate variance estimation, however for the purpose of this vignette we perform the analysis on 3 vs 3. In this example, we are interested in testing for differential variability in the cancer versus normal group. When we specify the *coef* parameter, which corresponds to the columns of the design matrix to be used for testing differential variability, we specify both the intercept and the fourth column. The ID variable is a nuisance parameter and not used when obtaining the absolute deviations, however it can be included in the linear modelling step. For methylation data, the `varFit` function will take either a matrix of M-values, β values or a *MethylSet* object as input. If β values are supplied, a logit transformation is performed. Note that as a default, `varFit` uses the robust setting in the *limma* framework, which requires the use of the *statmod* package.

```
fitvar <- varFit(Mval, design = design, coef = c(1,4))
```

The numbers of hyper-variable (1) and hypo-variable (-1) genes in cancer vs normal can be obtained using `decideTests`.

```
summary(decideTests(fitvar))
```

```
##    (Intercept) idid2 idid3 groupcancer
## -1           0     3     3           0
## 0        19739 19994 19988       19996
## 1          261     3     9           4
```

```
topDV <- topVar(fitvar, coef=4)
topDV
```

```
##            SampleVar LogVarRatio DiffLevene        t      P.Value  Adj.P.Value
## cg11140785  7.543025    3.604785   3.067140 5.912416 3.398560e-09 6.797119e-05
## cg00691999  6.555713    3.334991   2.707965 4.953892 7.303999e-07 7.303999e-03
## cg25903779  5.529399    4.248514   2.741791 4.599897 4.239944e-06 2.826629e-02
## cg17942639  5.095561    4.055017   2.669105 4.471327 7.794920e-06 3.897460e-02
## cg04048250  4.305286    3.723866   2.610678 4.358441 1.313187e-05 5.252747e-02
## cg16779463  6.375230    4.341511   2.614586 4.218080 2.469331e-05 7.760951e-02
## cg18235000  4.094527    3.771334   2.526205 4.196519 2.716333e-05 7.760951e-02
## cg25230111  4.230815    4.634277   2.596252 4.073351 4.643066e-05 1.064744e-01
## cg15063355  6.412194    8.097028   2.711108 4.057999 4.958882e-05 1.064744e-01
```

```
## cg19572637  4.660908    5.838573   2.642029 4.018646 5.864065e-05 1.064744e-01
```

An alternate parameterisation of the design matrix that does not include an intercept term can also be used, and specific contrasts tested with `contrasts.varFit`. Here we specify the design matrix such that the first two columns correspond to the normal and cancer groups, respectively.

```
design2 <- model.matrix(~0+group+id)
fitvar.contr <- varFit(Mval, design=design2, coef=c(1,2))
contr <- makeContrasts(groupcancer-groupnormal,levels=colnames(design2))
fitvar.contr <- contrasts.varFit(fitvar.contr,contrasts=contr)
```

The results are identical to before.

```
summary(decideTests(fitvar.contr))
```

```
##    groupcancer - groupnormal
## -1                         0
## 0                      19996
## 1                          4
```

```
topVar(fitvar.contr,coef=1)
```

```
##            SampleVar LogVarRatio DiffLevene        t      P.Value  Adj.P.Value
## cg11140785  7.543025    3.604785   3.067140 5.912416 3.398560e-09 6.797119e-05
## cg00691999  6.555713    3.334991   2.707965 4.953892 7.303999e-07 7.303999e-03
## cg25903779  5.529399    4.248514   2.741791 4.599897 4.239944e-06 2.826629e-02
## cg17942639  5.095561    4.055017   2.669105 4.471327 7.794920e-06 3.897460e-02
## cg04048250  4.305286    3.723866   2.610678 4.358441 1.313187e-05 5.252747e-02
## cg16779463  6.375230    4.341511   2.614586 4.218080 2.469331e-05 7.760951e-02
## cg18235000  4.094527    3.771334   2.526205 4.196519 2.716333e-05 7.760951e-02
## cg25230111  4.230815    4.634277   2.596252 4.073351 4.643066e-05 1.064744e-01
## cg15063355  6.412194    8.097028   2.711108 4.057999 4.958882e-05 1.064744e-01
## cg19572637  4.660908    5.838573   2.642029 4.018646 5.864065e-05 1.064744e-01
```

The β values for the top 4 differentially variable CpGs can be seen in Figure 4.

8.2 RNA-Seq expression data

Testing for differential variability in expression data is straightforward if the technology is gene expression microarrays. The matrix of expression values can be supplied directly to the `varFit` function. For RNA-Seq data, the mean-variance relationship that occurs in count data needs to be taken into account. In order to deal with this issue, we apply a `voom` transformation [8] to obtain observation weights, which are then used in the linear modelling step. For RNA-Seq data, the `varFit` function will take a *DGEList* object as input.

To demonstrate this, we use data from the *tweeDEseqCountData* package. This data is part of the International HapMap project, consisting of RNA-Seq profiles from 69 unrelated Nigerian individuals [11]. The only covariate is gender, so we can look at differentially variable expression between males and females. We follow the code from the *limma* vignette to read in and process the data before testing for differential variability.

First we load up the data and extract the relevant information.

```
library(tweeDEseqCountData)
data(pickrell1)
counts<-exprs(pickrell1.eset)
dim(counts)
```

```
## [1] 38415    69
```

```
cpgsDV <- rownames(topDV)
par(mfrow=c(2,2))
for(i in 1:4){
stripchart(beta[rownames(beta)==cpgsDV[i],]~design[,4],method="jitter",
group.names=c("Normal","Cancer"),pch=16,cex=1.5,col=c(4,2),ylab="Beta values",
vertical=TRUE,cex.axis=1.5,cex.lab=1.5)
title(cpgsDV[i],cex.main=1.5)
}
```

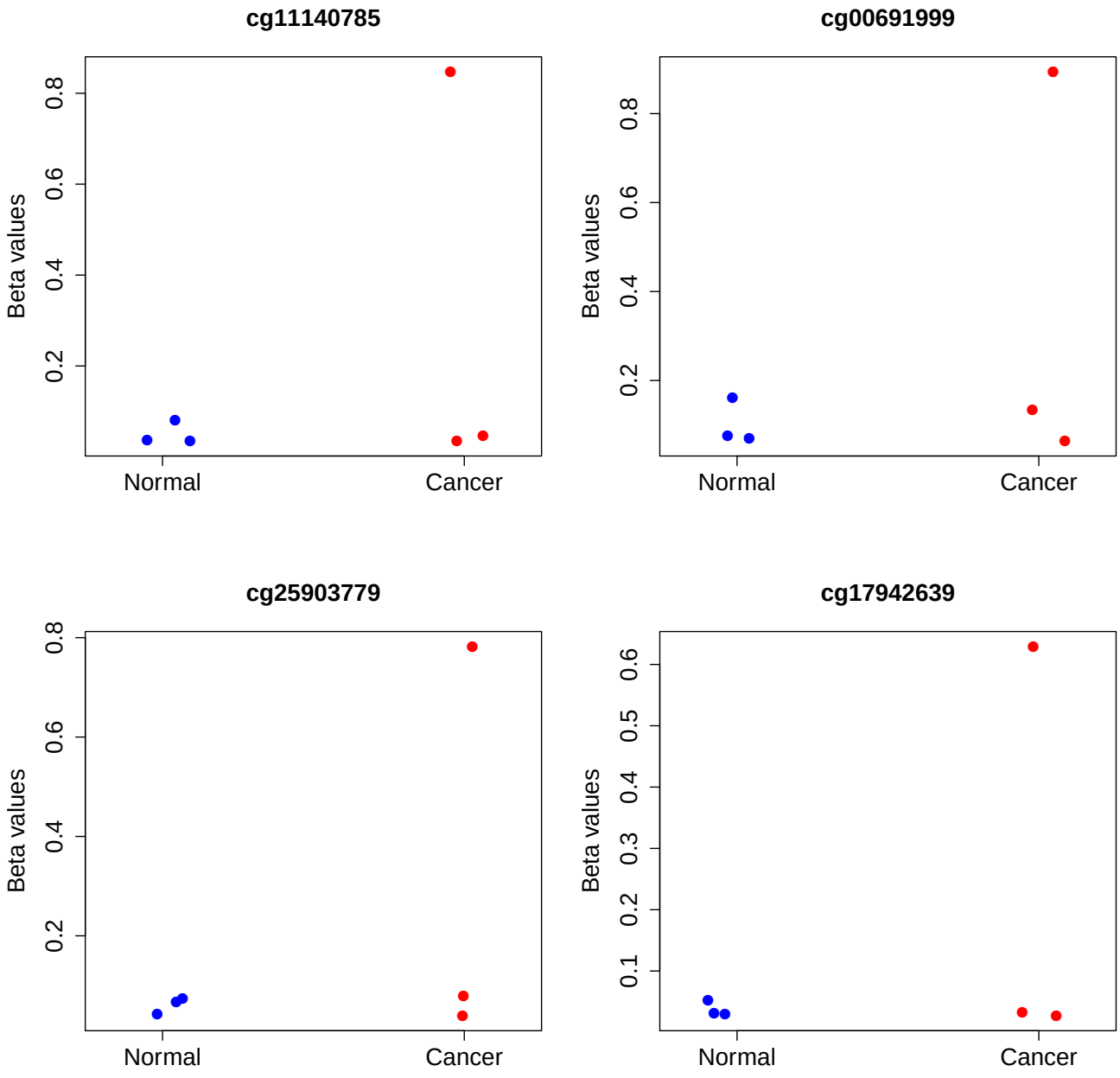

Figure 4: The β values for the top 4 differentially variable CpGs.

```
gender <- pickrell1.eset$gender
table(gender)
```

```
## gender
## female   male
##     40     29
```

```
rm(pickrell1.eset)
data(genderGenes)
data(annotEnsembl63)
annot <- annotEnsembl63[,c("Symbol","Chr")]
rm(annotEnsembl63)
```

We now have the counts, gender of each sample and annotation (gene symbol and chromosome) for each Ensemble gene. We can form a *DGEList* object using the *edgeR* package.

```
library(edgeR)
y <- DGEList(counts=counts, genes=annot[rownames(counts),])
```

We filter out lowly expressed genes by keeping genes with at least 1 count per million reads in at least 20 samples, as well as genes that have defined annotation. Finally we perform scaling normalisation.

```
isexpr <- rowSums(cpm(y)>1) >= 20
hasannot <- rowSums(is.na(y$genes))==0
y <- y[isexpr & hasannot,,keep.lib.sizes=FALSE]
dim(y)
```

```
## [1] 17310    69
```

```
y <- calcNormFactors(y)
```

We set up the design matrix and test for differential variability. In this case there are no nuisance parameters, so *coef* does not need to be explicitly specified.

```
design.hapmap <- model.matrix(~gender)
fitvar.hapmap <- varFit(y, design = design.hapmap)
```

```
## Converting counts to log counts-per-million using voom.
```

```
fitvar.hapmap$genes <- y$genes
```

We can display the results of the test:

```
summary(decideTests(fitvar.hapmap))
```

```
##    (Intercept) gendermale
## -1           0          2
## 0            0      17308
## 1        17310          0
```

```
topDV.hapmap <- topVar(fitvar.hapmap,coef=ncol(design.hapmap))
topDV.hapmap
```

```
##                       Symbol Chr  SampleVar LogVarRatio DiffLevene         t      P.Value
## ENSG00000213318 RP11-331F4.1  16 5.69839463   -2.562939 -0.9859943 -8.031075 7.238901e-12
## ENSG00000129824       RPS4Y1   Y 2.32497726   -2.087025 -0.4585620 -4.957078 3.959119e-06
## ENSG00000233864       TTTY15   Y 6.79004140   -2.245369 -0.6085233 -4.612613 1.497880e-05
## ENSG00000176171        BNIP3  10 0.41317384    1.199292  0.3632133  4.219459 6.439806e-05
## ENSG00000197358      BNIP3P1  14 0.39969125    1.149754  0.3353288  4.058182 1.147653e-04
## ENSG00000025039        RRAGD   6 0.91837213    1.091229  0.4926839  3.977116 1.527080e-04
## ENSG00000103671        TRIP4  15 0.07456448   -1.457139 -0.1520583 -3.911199 1.921645e-04
```

```
## ENSG00000171100         MTM1   X 0.44049558   -1.133295 -0.3334619 -3.896210 2.024119e-04
## ENSG00000149476          DAK  11 0.13289523   -1.470460 -0.1919880 -3.785886 2.956246e-04
## ENSG00000064886       CHI3L2   1 2.70234584    1.468059  0.8449434  3.782149 2.994084e-04
##                    Adj.P.Value
## ENSG00000213318 1.253054e-07
## ENSG00000129824 3.426618e-02
## ENSG00000233864 8.642769e-02
## ENSG00000176171 2.786826e-01
## ENSG00000197358 3.973173e-01
## ENSG00000025039 4.379688e-01
## ENSG00000103671 4.379688e-01
## ENSG00000171100 4.379688e-01
## ENSG00000149476 4.428665e-01
## ENSG00000064886 4.428665e-01
```

The log counts per million for the top 4 differentially variable genes can be seen in Figure 5.

9 Gene ontology analysis

Once a differential methylation or differential variability analysis has been performed, it may be of interest to know which gene pathways are targeted by the significant CpG sites. It is not entirely clear from the literature how best to perform such an analysis, however Geeleher et al. (2013) showed there is a severe bias when performing gene ontology analysis with methylation data. This is due to the fact that there are differing numbers of probes per gene on several different array technologies. For the Illumina Infinium HumanMethylation450 array the number of probes per gene ranges from 1 to 1299, with a median of 15 probes per gene. This means that when mapping CpG sites to genes, a gene is more likely to be "selected" if there are many CpG sites associated with the gene.

One way to take into account this selection bias is to model the relationship between the number of probes per gene and the probability of being selected. This can be performed by adapting the *goseq* method of Young et al. (2010). Each gene then has a prior probability associated with it, and a modified version of a hypergeometric test can be performed, testing for over-representation of the selected genes in each GO category. The function `gometh` performs this analysis, taking as input the list of significant CpGs.

To illustrate how to use `gometh`, consider the results from the differential methylation analysis with ruv.

```
topRUV(rfit2)
```

```
##            coefficients         t           p      p.BH     p.ebayes  p.ebayes.BH
## cg07155336     5.769286 15.345069 0.002005546 0.3431163 1.434834e-55 6.966293e-50
## cg06463958     5.733093 15.434797 0.001978272 0.3431163 6.749298e-55 1.638433e-49
## cg00024472     5.662959 15.946200 0.001832444 0.3431163 1.319390e-53 2.135266e-48
## cg02040433     5.651399 10.054445 0.005389436 0.3431163 2.146210e-53 2.605027e-48
## cg13355248     5.595396  9.963702 0.005504213 0.3431163 2.234891e-52 2.022589e-47
## cg02467990     5.592707  6.859614 0.013008521 0.3431163 2.499534e-52 2.022589e-47
## cg00817367     5.527501 13.070583 0.002921656 0.3431163 3.710480e-51 2.573547e-46
## cg11396157     5.487992 10.931263 0.004436178 0.3431163 1.873636e-50 1.137091e-45
## cg16306898     5.466780  5.573935 0.020790127 0.3431163 4.448085e-50 2.399554e-45
## cg03735888     5.396242 15.482605 0.001963955 0.3431163 7.700032e-49 3.738458e-44
```

```
table(rfit2$p.ebayes.BH < 0.01)
```

```
##
##  FALSE   TRUE
## 424168  61344
```

At a 1% false discovery rate cut-off, there are still tens of thousands of CpG sites differentially methylated. These will

```
genesDV <- rownames(topDV.hapmap)
par(mfrow=c(2,2))
for(i in 1:4){
stripchart(cpm(y,log=TRUE)[rownames(y)==genesDV[i],]~design.hapmap[,ncol(design.hapmap)],method="jitter",
group.names=c("Female","Male"),pch=16,cex=1.5,col=c(4,2),ylab="Log counts per million",
vertical=TRUE,cex.axis=1.5,cex.lab=1.5)
title(genesDV[i],cex.main=1.5)
}
```

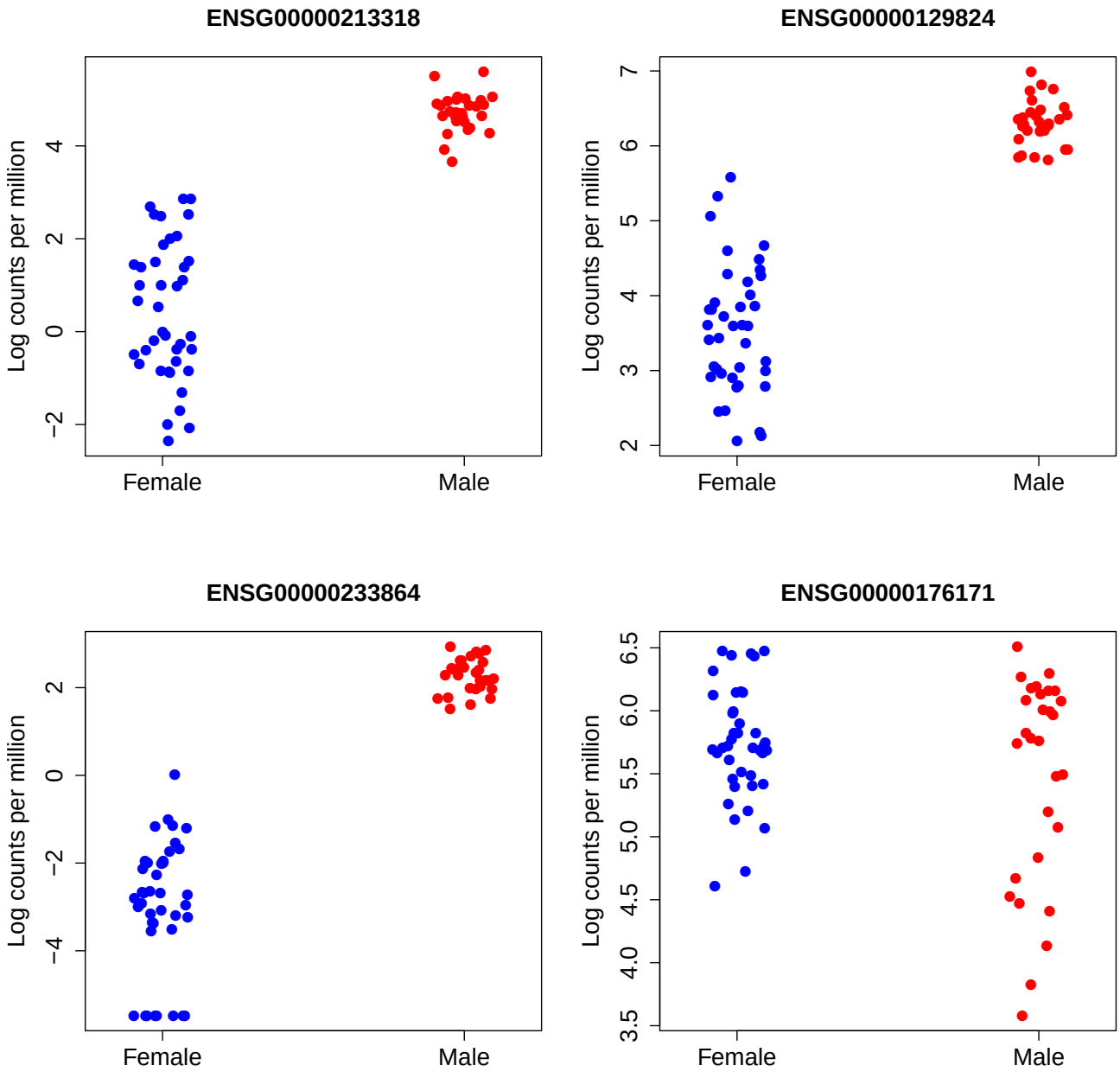

Figure 5: The log counts per million for the top 4 differentially variably expressed genes.

undoubtably map to almost all the genes in the genome, making a gene ontology analysis irrelevant. One option for selecting CpGs in this context is to apply not only a false discovery rate cut-off, but also a $\delta\beta$ cut-off. However, for this dataset, taking a relatively large $\delta\beta$ cut-off of 0.25 still leaves more than 30,000 CpGs differentially methylated.

```
beta <- getBeta(mSet)
beta_norm <- rowMeans(beta[,des[,2]==0])
beta_can <- rowMeans(beta[,des[,2]==1])
delta_beta <- beta_can - beta_norm
sigDM <- rfit2$p.ebayes.BH < 0.01 & abs(delta_beta) > 0.25
table(sigDM)
```

```
## sigDM
##  FALSE   TRUE
## 451760  33748
```

Instead, we take the top 10,000 CpG sites as input to `gometh`.

```
topCpGs<-topRUV(rfit2,number=10000)
sigCpGs <- rownames(topCpGs)
sigCpGs[1:10]
```

```
##  [1] "cg07155336" "cg06463958" "cg00024472" "cg02040433" "cg13355248" "cg02467990"
##  [7] "cg00817367" "cg11396157" "cg16306898" "cg03735888"
```

The `gometh` takes as input a character vector of CpG names, and optionally, a character vector of all CpG sites tested. If the *all.cpg* argument is omitted, all the CpGs on the array are used as background. If the *plot.bias* argument is TRUE, a figure showing the relationship between the probability of being selected and the number of probes per gene will be displayed. The function `topGO` shows the top enriched GO categories.

```
gst <- gometh(sig.cpg=sigCpGs, all.cpg=rownames(rfit2))
```

```
## Warning in alias2SymbolTable(flat$symbol): Multiple symbols ignored for one or more aliases
```

```
topGO(gst)
```

```
##                                          Term Ont    N   DE         P.DE          FDR
## GO:0048731                  system development  BP 3936  854 1.152581e-37 2.248109e-33
## GO:0007399           nervous system development  BP 2017  533 1.515388e-34 1.073083e-30
## GO:0007275 multicellular organismal development  BP 4501  925 1.650474e-34 1.073083e-30
## GO:0044707 single-multicellular organism process  BP 6259 1131 3.514819e-33 1.713914e-29
## GO:0032501      multicellular organismal process  BP 6496 1156 1.251299e-32 4.881316e-29
## GO:0048856      anatomical structure development  BP 4717  943 5.247702e-31 1.705941e-27
## GO:0005886                        plasma membrane  CC 4608  855 7.663540e-30 2.135391e-26
## GO:0044767 single-organism developmental process  BP 5233 1005 3.295872e-29 8.035749e-26
## GO:0071944                         cell periphery  CC 4697  863 2.734018e-28 5.925225e-25
## GO:0032502               developmental process  BP 5319 1012 3.264554e-28 5.940906e-25
## GO:0048513                   organ development  BP 2846  626 3.350421e-28 5.940906e-25
## GO:0048699               generation of neurons  BP 1302  373 1.753117e-27 2.849546e-24
## GO:0022008                        neurogenesis  BP 1378  387 2.579736e-27 3.870596e-24
## GO:0030182              neuron differentiation  BP 1196  348 1.108928e-26 1.544974e-23
## GO:0030154                cell differentiation  BP 3426  717 2.146010e-26 2.790528e-23
## GO:0007267                 cell-cell signaling  BP 1204  316 1.909087e-24 2.327296e-21
## GO:0007417    central nervous system development  BP  835  257 2.114265e-24 2.425808e-21
## GO:0009653    anatomical structure morphogenesis  BP 2447  563 3.364880e-24 3.646221e-21
## GO:0048869        cellular developmental process  BP 3643  738 6.010190e-24 6.169934e-21
## GO:0048468                     cell development  BP 1910  464 2.582545e-23 2.518627e-20
```

10 Session information

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

- R version 3.2.0 RC (2015-04-09 r68169), x86_64-apple-darwin10.8.0
- Locale: en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
- Base packages: base, datasets, graphics, grDevices, methods, parallel, stats, stats4, utils
- Other packages: AnnotationDbi 1.31.0, Biobase 2.29.0, BiocGenerics 0.15.0, Biostrings 2.37.0, bumphunter 1.9.0, DBI 0.3.1, edgeR 3.11.0, foreach 1.4.2, GenomeInfoDb 1.5.0, GenomicRanges 1.21.0, GO.db 3.1.2, IlluminaHumanMethylation450kanno.ilmn12.hg19 0.2.1, IlluminaHumanMethylation450kmanifest 0.4.0, IRanges 2.3.0, iterators 1.0.7, lattice 0.20-31, limma 3.25.0, locfit 1.5-9.1, minfi 1.15.0, minfiData 0.9.1, missMethyl 1.3.0, org.Hs.eg.db 3.1.2, RSQLite 1.0.0, S4Vectors 0.7.0, tweeDEseqCountData 1.5.1, XVector 0.9.0
- Loaded via a namespace (and not attached): annotate 1.47.0, base64 1.1, beanplot 1.2, BiasedUrn 1.06.1, BiocParallel 1.3.0, BiocStyle 1.7.0, biomaRt 2.25.0, bitops 1.0-6, codetools 0.2-11, colorspace 1.2-6, digest 0.6.8, doRNG 1.6, evaluate 0.6, formatR 1.1, futile.logger 1.4, futile.options 1.0.0, genefilter 1.51.0, GenomicAlignments 1.5.0, GenomicFeatures 1.21.0, GEOquery 2.35.0, grid 3.2.0, highr 0.4.1, illuminaio 0.11.0, knitr 1.9, lambda.r 1.1.7, MASS 7.3-40, matrixStats 0.14.0, mclust 5.0.0, methylumi 2.15.0, multtest 2.25.0, nlme 3.1-120, nor1mix 1.2-0, pkgmaker 0.22, plyr 1.8.1, preprocessCore 1.31.0, quadprog 1.5-5, RColorBrewer 1.1-2, Rcpp 0.11.5, RCurl 1.95-4.5, registry 0.2, reshape 0.8.5, rngtools 1.2.4, Rsamtools 1.21.0, rtracklayer 1.29.1, ruv 0.9.4, siggenes 1.43.0, splines 3.2.0, statmod 1.4.21, stringr 0.6.2, survival 2.38-1, tools 3.2.0, XML 3.98-1.1, xtable 1.7-4, zlibbioc 1.15.0

References

[1] Aryee, M. J., Jaffe, A. E., Corrada-Bravo, H., Ladd-Acosta, C., Feinberg, A. P., Hansen, K. D., and Irizarry, R. A. (2014). Minfi: a flexible and comprehensive Bioconductor package for the analysis of Infinium DNA methylation microarrays. *Bioinformatics*, **30**(10):1363-1369.

[2] Bibikova, M., Barnes, B., Tsan, C., Ho, V., Klotzle, B, Le, J.M., Delano, D., Zhang, L., Schroth, G.P., Gunderson, K.L., Fan, J., and Shen, R. (2011). High density DNA methylation array with single CpG site resolution. *Genomics*, **98**(4):288-295.

[3] Dedeurwaerder, S., Defrance, M., Calonne, E., Denis, H., Sotiriou, C., and Fuks, F. (2011). Evaluation of the Infinium Methylation 450K technology. *Epigenomics*, **3**(6):771-784.

[4] Gagnon-Bartsch JA, Speed TP. (2012). Using control genes to correct for unwanted variation in microarray data. *Biostatistics*. **13**(3), 539-52. Available at: http://biostatistics.oxfordjournals.org/content/13/3/539.full.

[5] Gagnon-Bartsch, Jacob, and Speed. (2013). Removing Unwanted Variation from High Dimensional Data with Negative Controls. Available at: http://statistics.berkeley.edu/tech-reports/820.

[6] Geeleher, P., Hartnett, L., Egan, L. J., Golden, A., Ali, R. A. R., and Seoighe, C. (2013). Gene-set analysis is severely biased when applied to genome-wide methylation data. *Bioinformatics*, **29**(15), 1851-1857.

[7] Hansen, K.D., Timp, W., Bravo, H.C., Sabunciyan, S., Langmead, B., McDonald, O.G., Wen, B., Wu, H., Liu, Y., Diep, D., Briem, E., Zhang, K., Irizarry, R., and Feinberg, A.P. (2011). Increased methylation variation in epigenetic domains across cancer types. *Nature Genetics*, **43**:768-75.

[8] Law, C.W., Chen, Y., Shi, W., Smyth, G.K. (2014). Voom: precision weights unlock linear model analysis tools for RNA-seq read counts. *Genome Biology*, 15, R29.

[9] Maksimovic, J., Gordon, L., and Oshlack, A. (2012). SWAN: Subset-quantile within array normalization for illumina infinium HumanMethylation450 BeadChips. *Genome Biology*, **13**(6):R44.

[10] Phipson, B., and Oshlack, A. DiffVar: A new method for detecting differential variability with application to methylation in cancer and ageing. *Genome Biology, Accepted 10 September 2014*.

[11] Pickrell, J.K., Marioni, J.C., Pai, A.A., Degner, J.F., Engelhardt, B.E., Nkadori, E., Veyrieras, J.B., Stephens, M., Gilad, Y., and Pritchard, J.K. (2010). Understanding mechanisms underlying human gene expression variation with RNA sequencing. *Nature*, 464, 768-72.

[12] Smyth, G.K. (2005). Limma: linear models for microarray data. In: 'Bioinformatics and Computational Biology Solutions using R and Bioconductor'. R. Gentleman, V. Carey, S. Dudoit, R. Irizarry, W. Huber (eds), Springer, New York, pages 397-420.

[13] Young, M. D., Wakefield, M. J., Smyth, G. K., and Oshlack, A. (2010). Gene ontology analysis for RNA-seq: accounting for selection bias. *Genome Biology*, 11, R14.