Methylation HW11

Jessica Murphy

May 6, 2019

```
suppressWarnings(suppressMessages(library("minfi", quietly=T)))
suppressWarnings(suppressMessages(library("wateRmelon", quietly=T)))
suppressWarnings(suppressMessages(library("ChAMP", quietly=T)))
suppressWarnings(suppressMessages(library("sva", quietly=T)))
suppressWarnings(suppressMessages(library("RColorBrewer", quietly=T)))

sessionInfo()

## R version 3.5.1 (2018-07-02)
## Platform: x86_64-redhat-linux-gnu (64-bit)
## Running under: CentOS release 6.10 (Final)
##
## Matrix products: default
## BLAS: /usr/lib64/R/lib/libRblas.so
## LAPACK: /usr/lib64/R/lib/libRlapack.so
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=en_US.UTF-8        LC_COLLATE=en_US.UTF-8
##  [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
##  [1] splines   stats4    parallel  stats     graphics  grDevices utils
##  [8] datasets  methods   base
##
## other attached packages:
##  [1] RColorBrewer_1.1-2
##  [2] sva_3.30.1
##  [3] genefilter_1.64.0
##  [4] mgcv_1.8-27
##  [5] nlme_3.1-137
##  [6] ChAMP_2.12.4
##  [7] IlluminaHumanMethylationEPICmanifest_0.3.0
##  [8] Illumina450ProbeVariants.db_1.18.0
##  [9] DMRcate_1.18.0
## [10] DMRcatedata_1.18.0
## [11] DSS_2.30.1
## [12] bsseq_1.18.0
## [13] FEM_3.10.0
## [14] graph_1.60.0
## [15] impute_1.56.0
## [16] igraph_1.2.4
## [17] corrplot_0.84
## [18] marray_1.60.0
## [19] Matrix_1.2-15
## [20] ChAMPdata_2.14.1
## [21] wateRmelon_1.26.0
## [22] illuminaio_0.24.0
## [23] IlluminaHumanMethylation450kanno.ilmn12.hg19_0.6.0
## [24] ROC_1.58.0
## [25] lumi_2.34.0
## [26] methylumi_2.28.0
## [27] FDb.InfiniumMethylation.hg19_2.2.0
## [28] org.Hs.eg.db_3.7.0
## [29] TxDb.Hsapiens.UCSC.hg19.knownGene_3.2.2
## [30] GenomicFeatures_1.34.3
## [31] AnnotationDbi_1.44.0
## [32] ggplot2_3.1.0
## [33] reshape2_1.4.3
## [34] scales_1.0.0
## [35] limma_3.38.3
## [36] minfi_1.28.3
## [37] bumphunter_1.24.5
## [38] locfit_1.5-9.1
## [39] iterators_1.0.10
## [40] foreach_1.4.4
## [41] Biostrings_2.50.2
## [42] XVector_0.22.0
## [43] SummarizedExperiment_1.12.0
## [44] DelayedArray_0.8.0
## [45] BiocParallel_1.16.6
## [46] matrixStats_0.54.0
## [47] Biobase_2.42.0
## [48] GenomicRanges_1.34.0
## [49] GenomeInfoDb_1.18.2
## [50] IRanges_2.16.0
## [51] S4Vectors_0.20.1
## [52] BiocGenerics_0.28.0
## [53] knitr_1.21
##
## loaded via a namespace (and not attached):
##   [1] rtracklayer_1.42.1
##   [2] prabclus_2.2-7
##   [3] R.methodsS3_1.7.1
##   [4] pkgmaker_0.27
##   [5] tidyr_0.8.2
##   [6] acepack_1.4.1
##   [7] bit64_0.9-7
##   [8] R.utils_2.8.0
##   [9] data.table_1.12.0
##  [10] rpart_4.1-13
##  [11] RCurl_1.95-4.11
##  [12] GEOquery_2.50.5
##  [13] AnnotationFilter_1.6.0
##  [14] doParallel_1.0.14
##  [15] preprocessCore_1.44.0
##  [16] RSQLite_2.1.1
##  [17] combinat_0.0-8
##  [18] bit_1.1-14
##  [19] xml2_1.2.0
##  [20] httpuv_1.4.5.1
##  [21] assertthat_0.2.0
##  [22] IlluminaHumanMethylation450kmanifest_0.4.0
##  [23] IlluminaHumanMethylationEPICanno.ilm10b4.hg19_0.6.0
##  [24] viridis_0.5.1
##  [25] isva_1.9
##  [26] xfun_0.5
##  [27] hms_0.4.2
##  [28] evaluate_0.13
##  [29] missMethyl_1.16.0
##  [30] DNAcopy_1.56.0
##  [31] promises_1.0.1
##  [32] DEoptimR_1.0-8
##  [33] progress_1.2.0
##  [34] dendextend_1.9.0
##  [35] DBI_1.0.0
##  [36] htmlwidgets_1.3
##  [37] reshape_0.8.8
##  [38] purrr_0.3.0
##  [39] dplyr_0.8.0.1
##  [40] backports_1.1.3
## [41] permute_0.9-4
## [42] trimcluster_0.1-2.1
## [43] annotate_1.60.0
## [44] biomaRt_2.38.0
## [45] ensembldb_2.6.6
## [46] withr_2.1.2
## [47] globaltest_5.36.0
## [48] Gviz_1.26.5
## [49] BSgenome_1.50.0
## [50] robustbase_0.93-3
## [51] checkmate_1.9.1
## [52] GenomicAlignments_1.18.1
## [53] prettyunits_1.0.2
## [54] mclust_5.4.2
## [55] cluster_2.0.7-1
## [56] RPMM_1.25
## [57] lazyeval_0.2.1
## [58] crayon_1.3.4
## [59] pkgconfig_2.0.2
## [60] ProtGenerics_1.14.0
## [61] nnet_7.3-12
## [62] rlang_0.3.1
## [63] diptest_0.75-7
## [64] nleqslv_3.3.2
## [65] registry_0.5
## [66] affyio_1.52.0
## [67] dichromat_2.0-0
## [68] rngtools_1.3.1
## [69] base64_2.0
## [70] Rhdf5lib_1.4.2
## [71] base64enc_0.1-3
## [72] geneLenDataBase_1.18.0
## [73] whisker_0.3-2
## [74] viridisLite_0.3.0
## [75] bitops_1.0-6
## [76] R.oo_1.22.0
## [77] KernSmooth_2.23-15
## [78] blob_1.1.1
## [79] DelayedMatrixStats_1.4.0
## [80] doRNG_1.7.1
## [81] stringr_1.4.0
## [82] qvalue_2.14.1
## [83] nor1mix_1.2-3
## [84] readr_1.3.1
## [85] memoise_1.1.0
##  [86] magrittr_1.5
##  [87] plyr_1.8.4
##  [88] bibtex_0.4.2
##  [89] zlibbioc_1.28.0
##  [90] compiler_3.5.1
##  [91] clue_0.3-56
##  [92] Rsamtools_1.34.1
##  [93] affy_1.60.0
##  [94] JADE_2.0-1
##  [95] IlluminaHumanMethylationEPICanno.ilm10b2.hg19_0.6.0
##  [96] htmlTable_1.13.1
##  [97] Formula_1.2-3
##  [98] MASS_7.3-51.1
##  [99] tidyselect_0.2.5
## [100] stringi_1.3.1
## [101] askpass_1.1
## [102] latticeExtra_0.6-28
## [103] grid_3.5.1
## [104] VariantAnnotation_1.28.11
## [105] tools_3.5.1
## [106] ruv_0.9.7
## [107] rstudioapi_0.9.0
## [108] foreign_0.8-71
## [109] gridExtra_2.3
## [110] digest_0.6.18
## [111] BiocManager_1.30.4
## [112] shiny_1.2.0
## [113] quadprog_1.5-5
## [114] fpc_2.1-11.1
## [115] Rcpp_1.0.0
## [116] siggenes_1.56.0
## [117] later_0.8.0
## [118] httr_1.4.0
## [119] biovizBase_1.30.1
## [120] kernlab_0.9-27
## [121] colorspace_1.4-0
## [122] XML_3.98-1.17
## [123] statmod_1.4.30
## [124] kpmt_0.1.0
## [125] multtest_2.38.0
## [126] shinythemes_1.1.2
## [127] flexmix_2.3-15
## [128] plotly_4.8.0
## [129] xtable_1.8-3
## [130] jsonlite_1.6
## [131] modeltools_0.2-22
## [132] R6_2.4.0
## [133] Hmisc_4.2-0
## [134] pillar_1.3.1
## [135] htmltools_0.3.6
## [136] mime_0.6
## [137] glue_1.3.0
## [138] class_7.3-15
## [139] beanplot_1.2
## [140] codetools_0.2-16
## [141] mvtnorm_1.0-8
## [142] lattice_0.20-38
## [143] tibble_2.0.1
## [144] curl_3.3
## [145] BiasedUrn_1.07
## [146] gtools_3.8.1
## [147] GO.db_3.7.0
## [148] openssl_1.2.1
## [149] survival_2.43-3
## [150] rmarkdown_1.11
## [151] fastICA_1.2-1
## [152] munsell_0.5.0
## [153] rhdf5_2.26.2
## [154] GenomeInfoDbData_1.2.0
## [155] goseq_1.34.1
## [156] HDF5Array_1.10.1
## [157] gtable_0.2.0
```

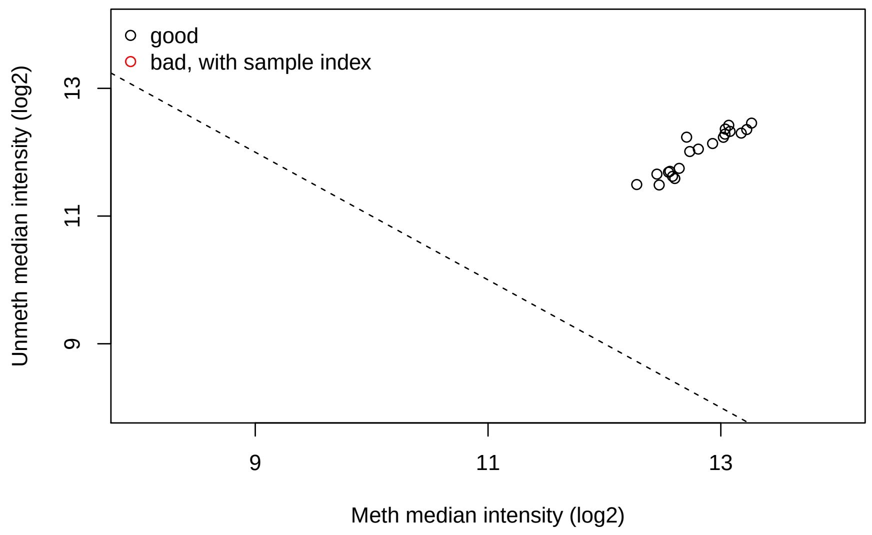
1 Read Data

```
baseDir1 = "/BIOS6660/Methylation/plate1" #remove tildas before knitting
targets1 = read.metharray.sheet(baseDir1)

## [read.metharray.sheet] Found the following CSV files:

## [1] "/BIOS6660/Methylation/plate1/selected_plate1.csv"

baseDir2 = "/BIOS6660/Methylation/plate2"
targets2 = read.metharray.sheet(baseDir2)

## [read.metharray.sheet] Found the following CSV files:

## [1] "/BIOS6660/Methylation/plate2/selected_plate2.csv"
```

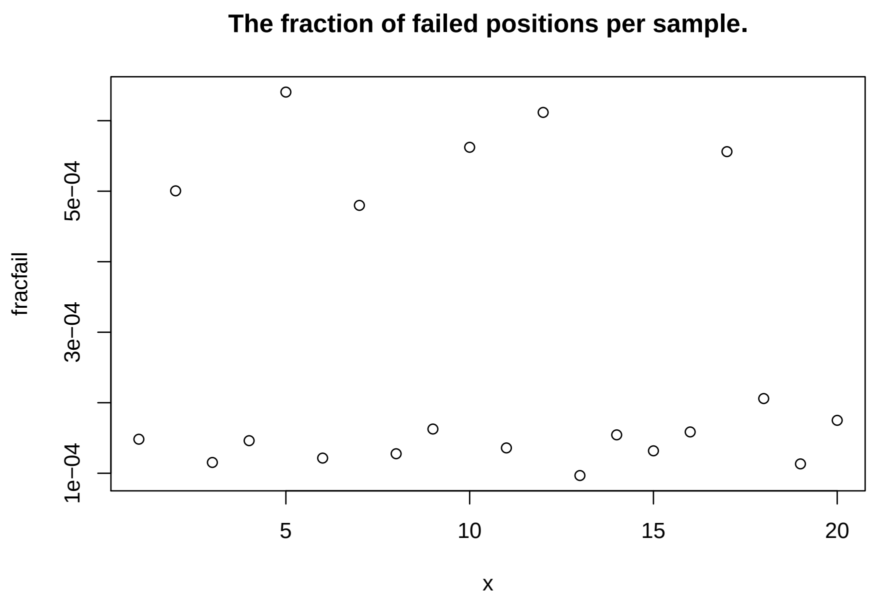
```
targets = rbind(targets1, targets2)

rgSet = read.metharray.exp(targets=targets, extended=T)
sampleNames(rgSet) = rgSet[[1]]
getManifest(rgSet)

## Loading required package:  IlluminaHumanMethylation450kmanifest

## IlluminaMethylationManifest object
## Annotation
##   array: IlluminaHumanMethylation450k
## Number of type I probes: 135476
## Number of type II probes: 350036
## Number of control probes: 850
## Number of SNP type I probes: 25
## Number of SNP type II probes: 40

#For deconvolution, the reference panel data are not an object of RGChannelSetExtended
rgSet_d = read.metharray.exp(targets=targets, extended=F)
sampleNames(rgSet_d) = rgSet_d[[1]]
getManifest(rgSet_d)

## IlluminaMethylationManifest object
## Annotation
##   array: IlluminaHumanMethylation450k
## Number of type I probes: 135476
## Number of type II probes: 350036
## Number of control probes: 850
## Number of SNP type I probes: 25
## Number of SNP type II probes: 40

clindat = read.table("/BIOS6660/Methylation/demographic.txt",
                     sep="\t", header=T)
stopifnot(all(clindat$Sample_Name==rgSet$Sample_Name))
pData(rgSet)$Sample_Group = clindat$Exposure
pData(rgSet)$child_sex = clindat$child_sex
```

2 Detection P value

```
mset = preprocessRaw(rgSet)

qc = getQC(mset)
plotQC(qc)
```



```r
detP = detectionP(rgSet)
detPcut = 0.05

failed = detP > detPcut
fracfail = colMeans(failed)
main = paste("The fraction of failed positions per sample.")
x = seq(1, length(fracfail), 1)
plot(x, fracfail, main=main)
```



```
removeDetP = 0.1
badProbes = rowMeans(failed) > removeDetP

mset.f = mset[!badProbes,]
mset = mset.f

message("There are ", sum(badProbes),
        " bad probes with high detection P values removed.")

## There are 373 bad probes with high detection P values removed.
```

3 Check bead count

```
beadCutoff = 0.1
bc = beadcount(rgSet)
quantile(bc, na.rm=T)

##   0%  25%  50%  75% 100%
##    3   11   14   17  108

bc2 = bc[rowSums(is.na(bc)) < beadCutoff*(ncol(bc)), ]
mset.f2 = mset[featureNames(mset) %in% row.names(bc2), ]
```

```
message("Filtering probes with a beadcount <3 in at least ",
        beadCutoff*100, "% of samples has removed ", dim(mset)[1]-
          dim(mset.f2)[1], " from the analysis.")

## Filtering probes with a beadcount <3 in at least 10% of samples
has removed 578 from the analysis.

mset = mset.f2
```

4 Check non-CG probes

```
mset.cg = dropMethylationLoci(mset, dropCH=T)

message("There are ", dim(mset)[1]-dim(mset.cg)[1],
        " non-CG probes. Keep them in the final analysis dataset.")

## There are 3090 non-CG probes.  Keep them in the final analysis dataset.
```

5 Map to the genome

```
gset = mapToGenome(mset)

annotation = getAnnotation(gset, dropNonMapping=F)
names(annotation)

##  [1] "chr"                        "pos"
##  [3] "strand"                     "Name"
##  [5] "AddressA"                   "AddressB"
##  [7] "ProbeSeqA"                  "ProbeSeqB"
##  [9] "Type"                       "NextBase"
## [11] "Color"                      "Probe_rs"
## [13] "Probe_maf"                  "CpG_rs"
## [15] "CpG_maf"                    "SBE_rs"
## [17] "SBE_maf"                    "Islands_Name"
## [19] "Relation_to_Island"         "Forward_Sequence"
## [21] "SourceSeq"                  "Random_Loci"
## [23] "Methyl27_Loci"              "UCSC_RefGene_Name"
## [25] "UCSC_RefGene_Accession"     "UCSC_RefGene_Group"
## [27] "Phantom"                    "DMR"
## [29] "Enhancer"                   "HMM_Island"
## [31] "Regulatory_Feature_Name"    "Regulatory_Feature_Group"
## [33] "DHS"

table(annotation$chr)

##
##   chr1 chr10 chr11 chr12 chr13 chr14 chr15 chr16 chr17 chr18 chr19  chr2
## 46792 24360 28760 24497 12268 15053 15246 21941 27832  5915 25486 34769
## chr20 chr21 chr22  chr3  chr4  chr5  chr6  chr7  chr8  chr9  chrX  chrY
## 10363  4240  8526 25114 20433 24291 36523 29972 20915  9853 11216   196

dim(annotation)

## [1] 484561     33

annotation2 = getAnnotation(gset, dropNonMapping=T)
dim(annotation2)

## [1] 484561     33

message("There are ", dim(annotation)[1]-dim(annotation2)[1],
        " non-mapping probes.")

## There are 0 non-mapping probes.

autosomes = annotation[!annotation$chr %in% c("chrX", "chrY"), ]
allosomes = annotation[annotation$chr %in% c("chrX", "chrY"), ]
```

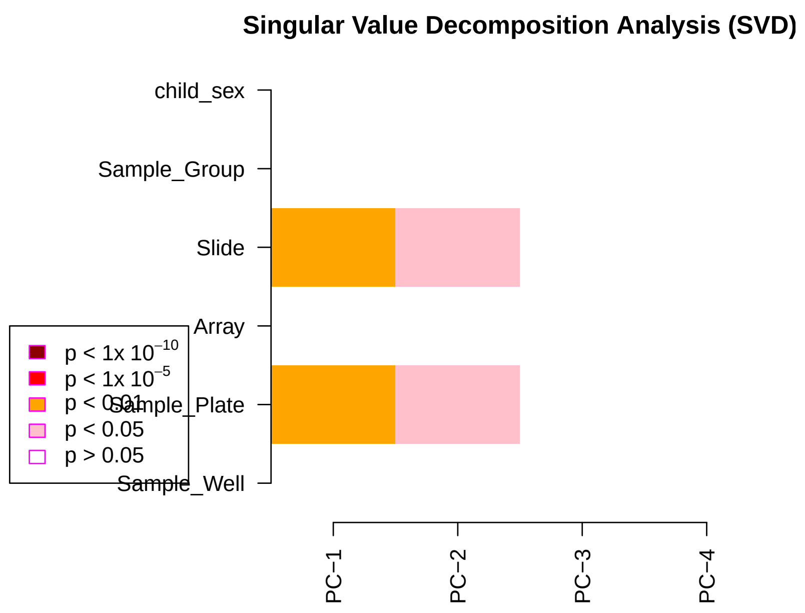
6 Identify probes with SNP

```
gset = addSnpInfo(gset)

getAnnotationObject(gset)

## IlluminaMethylationAnnotation object
## Annotation
##   array: IlluminaHumanMethylation450k
##   annotation: ilmn12
##   genomeBuild: hg19
## Available annotation
##   Islands.UCSC
##   Locations
##   Manifest
##   Other
##   SNPs.132CommonSingle
##   SNPs.135CommonSingle
##   SNPs.137CommonSingle
##   SNPs.138CommonSingle
##   SNPs.141CommonSingle
##   SNPs.142CommonSingle
##   SNPs.144CommonSingle
##   SNPs.146CommonSingle
##   SNPs.147CommonSingle
##   SNPs.Illumina
## Defaults
##   Locations
##   Manifest
##   SNPs.137CommonSingle
##   Islands.UCSC
##   Other

gset.f = dropLociWithSnps(gset, snps=c("SBE", "CpG"), maf=0)

message("The number of probes with snps is ", dim(gset)[1]-
          dim(gset.f)[1], ". Keep them for now.")

## The number of probes with snps is 17438.  Keep them for now.
```

7 Plot raw β and M values

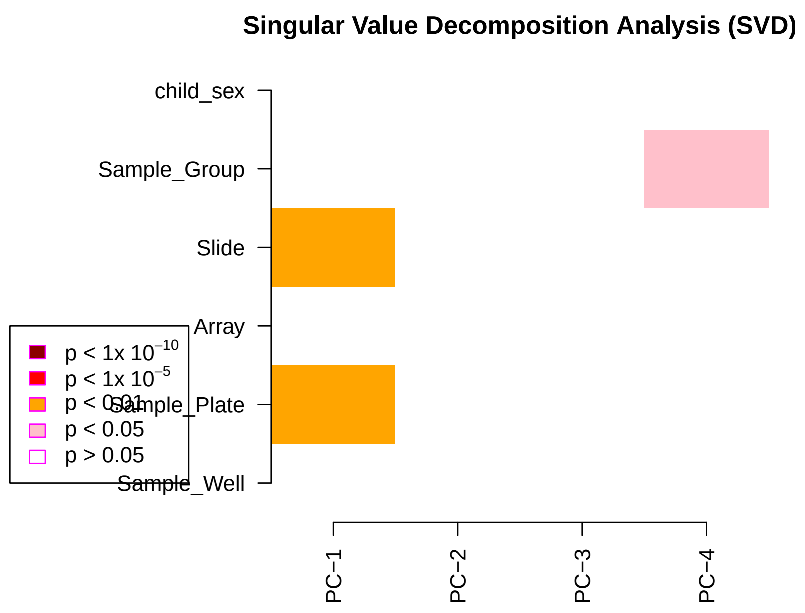
```
beta.raw = getBeta(gset)
M.raw = getM(gset)
colnames(beta.raw) = sampleNames(mset)
colnames(M.raw) = sampleNames(mset)

M.raw.auto = M.raw[rownames(M.raw) %in% rownames(autosomes),]
beta.raw.auto = beta.raw[rownames(beta.raw) %in% rownames(autosomes),]
colnames(beta.raw.auto) = sampleNames(mset)
colnames(M.raw.auto) = sampleNames(mset)

missing_names = rownames(which(is.na(beta.raw.auto), arr.ind = T))
champ.SVD(beta=beta.raw.auto[!rownames(beta.raw.auto) %in% missing_names, ],
          pd=pData(gset), resultsDir=paste(getwd(), "resultsChamp1", sep="/"))

## [===========================]
## [<<<< ChAMP.SVD START >>>>]
## ---------------
## champ.SVD Results will be saved in /home/murphjes/BIOS6660/Homework_11/resultsChamp1
.
```

```
## [SVD analysis will be proceed with 473114 probes and 20 samples.]
##
## [ champ.SVD() will only check the dimensions between data and pd,
instead if checking if Sample_Names are correctly matched (because
some user may have no Sample_Names in their pd file),thus please make
sure your pd file is in accord with your data sets (beta) and (rgSet).]
## « Following Factors in your pd(sample_sheet.csv) will be analysised:
»
## <Sample_Well>(character):D01, D02, D04, D05, D06, D07, D08, D10,
D11, D12, F01, F02, F04, F05, F06, F07, F08, F10, F11, F12
## <Sample_Plate>(character):Plate 1, Plate 2
## <Array>(character):R01C01, R02C01, R04C01, R05C01, R06C01, R01C02,
R02C02, R04C02, R05C02, R06C02
## <Slide>(character):9721366035, 9992576163
## <Sample_Group>(integer):1, 0
## <child_sex>(factor):M, F
## [champ.SVD have automatically select ALL factors contain at least
two different values from your pd(sample_sheet.csv), if you don't want
to analysis some of them, please remove them manually from your pd
variable then retry champ.SVD().]
##
## « Following Factors in your pd(sample_sheet.csv) will not be analysis:
»
## <Sample_Name>
## <Sample.Group>
## <Pool_ID>
## <Basename>
## <filenames>
## [Factors are ignored because they only indicate Name or Project,
or they contain ONLY ONE value across all Samples.]
##
## « PhenoTypes.lv generated successfully.  »
## « Calculate SVD matrix successfully.  »
## « Plot SVD matrix successfully.  »
## [««« ChAMP.SVD END »»»]
## [===========================]
## [If the batch effect is not significant, you may want to process
champ.DMP() or champ.DMR() or champ.BlockFinder() next, otherwise,
you may want to run champ.runCombat() to eliminat batch effect, then
rerun champ.SVD() to check corrected result.]
```

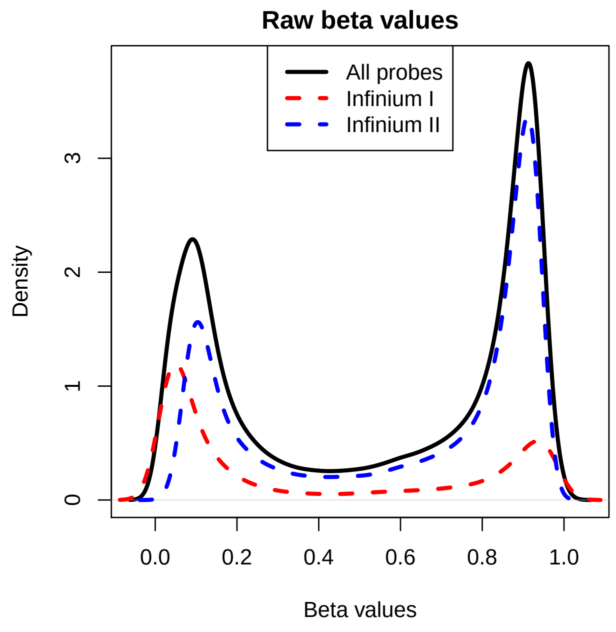


```
missing_names = rownames(which(is.na(M.raw.auto) | is.infinite(M.raw.auto),
                               arr.ind = T))
champ.SVD(beta=M.raw.auto[!rownames(M.raw.auto) %in% missing_names, ],
          pd=pData(gset), resultsDir=paste(getwd(), "resultsChamp1", sep="/"))
```

```
## [===========================]
## [<<<< ChAMP.SVD START >>>>>]
## ---------------
## champ.SVD Results will be saved in /home/murphjes/BIOS6660/Homework_11/resultsChamp1
.
## [SVD analysis will be proceed with 472873 probes and 20 samples.]
##
## [ champ.SVD() will only check the dimensions between data and pd,
instead if checking if Sample_Names are correctly matched (because
some user may have no Sample_Names in their pd file),thus please make
sure your pd file is in accord with your data sets (beta) and (rgSet).]
## << Following Factors in your pd(sample_sheet.csv) will be analysised:
>>
## <Sample_Well>(character):D01, D02, D04, D05, D06, D07, D08, D10,
D11, D12, F01, F02, F04, F05, F06, F07, F08, F10, F11, F12
## <Sample_Plate>(character):Plate 1, Plate 2
## <Array>(character):R01C01, R02C01, R04C01, R05C01, R06C01, R01C02,
R02C02, R04C02, R05C02, R06C02
## <Slide>(character):9721366035, 9992576163
## <Sample_Group>(integer):1, 0
## <child_sex>(factor):M, F
## [champ.SVD have automatically select ALL factors contain at least
two different values from your pd(sample_sheet.csv), if you don't want
to analysis some of them, please remove them manually from your pd
variable then retry champ.SVD().]
##
## « Following Factors in your pd(sample_sheet.csv) will not be analysis:
»
## <Sample_Name>
## <Sample.Group>
## <Pool_ID>
## <Basename>
## <filenames>
## [Factors are ignored because they only indicate Name or Project,
or they contain ONLY ONE value across all Samples.]
##
## « PhenoTypes.lv generated successfully.  »
## « Calculate SVD matrix successfully.  »
## « Plot SVD matrix successfully.  »
## [««« ChAMP.SVD END »»»]
## [===========================]
## [If the batch effect is not significant, you may want to process
champ.DMP() or champ.DMR() or champ.BlockFinder() next, otherwise,
you may want to run champ.runCombat() to eliminat batch effect, then
rerun champ.SVD() to check corrected result.]
```



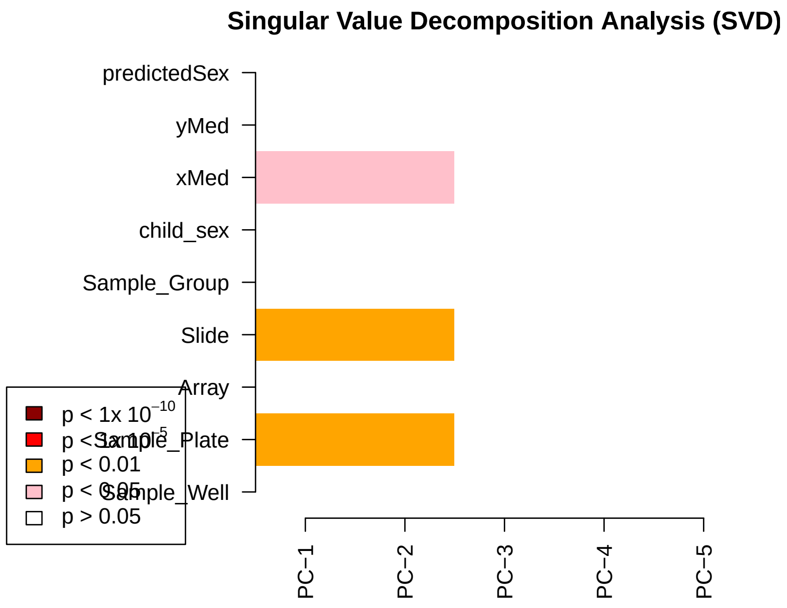
```
#Find probe types
probe_types = data.frame(cbind(rownames(gset), getProbeType(gset)))

#Names are required by plotBetasByType function
names(probe_types) = c("Name", "Type")

#plot density of beta value using average beta
plotBetasByType(rowMeans(beta.raw), probeTypes=probe_types,
                legendPos="top", colors=c("black", "red", "blue"),
                main="Raw beta values", lwd = 3, cex.legend = 1)
```



8 Normalization

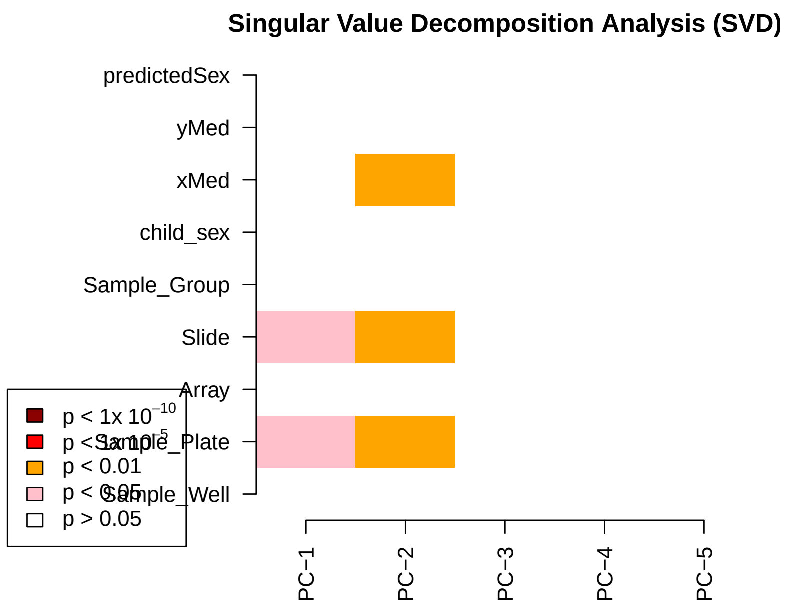
```
gset.norm = preprocessQuantile(gset, removeBadSamples = T)

## [preprocessQuantile] Mapping to genome.
## [preprocessQuantile] Fixing outliers.
## [preprocessQuantile] Quantile normalizing.

M.norm = getM(gset.norm)
beta.norm = getBeta(gset.norm)

colnames(beta.norm) = sampleNames(mset)
colnames(M.norm) = sampleNames(mset)

M.norm.auto = M.norm[rownames(M.norm) %in% rownames(autosomes),]
beta.norm.auto = beta.norm[rownames(beta.norm) %in% rownames(autosomes),]

colnames(beta.norm.auto) = sampleNames(mset)
colnames(M.norm.auto) = sampleNames(mset)

champ.SVD(beta=M.norm.auto,  pd=pData(gset.norm),
          resultsDir=paste(getwd(), "resultsChamp1", sep="/"))
```

```
## [===========================]
## [<<<< ChAMP.SVD START >>>>]
## ---------------
## champ.SVD Results will be saved in /home/murphjes/BIOS6660/Homework_11/resultsChamp1
.
## [SVD analysis will be proceed with 473149 probes and 20 samples.]
##
## [ champ.SVD() will only check the dimensions between data and pd,
instead if checking if Sample_Names are correctly matched (because
some user may have no Sample_Names in their pd file),thus please make
sure your pd file is in accord with your data sets (beta) and (rgSet).]
## << Following Factors in your pd(sample_sheet.csv) will be analysised:
>>
## <Sample_Well>(character):D01, D02, D04, D05, D06, D07, D08, D10,
D11, D12, F01, F02, F04, F05, F06, F07, F08, F10, F11, F12
## <Sample_Plate>(character):Plate 1, Plate 2
## <Array>(character):R01C01, R02C01, R04C01, R05C01, R06C01, R01C02,
R02C02, R04C02, R05C02, R06C02
## <Slide>(character):9721366035, 9992576163
## <Sample_Group>(integer):1, 0
## <child_sex>(factor):M, F
## <xMed>(numeric):13.4171932382483, 14.1502232804958, 13.47522677218,
13.6131563615466, 14.1760952119327, 13.2959841157302, 14.3023176188098,
13.5942079470926, 13.4799696797084, 14.2112799454771, 13.004922678569,
13.6373041221062, 13.313520782848, 13.3162107940661, 12.831109275357,
13.0134971542917, 13.5668270322379, 13.0176783665077, 13.0706241224303,
12.9999119395079
## <yMed>(numeric):13.7270655407971, 10.75359051341, 13.8279354058265,
13.9817453700246, 11.1617550090045, 13.6343004375137, 10.992913326597,
13.9655576614128, 13.829965645917, 10.9173651366522, 13.305624654563,
9.96000193206808, 13.6508801593559, 13.6613670707691, 13.2201801926979,
13.3667216684755, 9.89160384346522, 13.3788336362034, 13.4220647422222,
13.3565832195133
## <predictedSex>(character):M, F
## [champ.SVD have automatically select ALL factors contain at least
two different values from your pd(sample_sheet.csv), if you don't want
to analysis some of them, please remove them manually from your pd
variable then retry champ.SVD().]
##
## << Following Factors in your pd(sample_sheet.csv) will not be analysis:
>>
## <Sample_Name>
## <Sample.Group>
## <Pool_ID>
## <Basename>
## <filenames>
## [Factors are ignored because they only indicate Name or Project,
or they contain ONLY ONE value across all Samples.]
##
## « PhenoTypes.lv generated successfully.  »
## « Calculate SVD matrix successfully.  »
## « Plot SVD matrix successfully.  »
## [««« ChAMP.SVD END »»»]
## [===========================]
## [If the batch effect is not significant, you may want to process
champ.DMP() or champ.DMR() or champ.BlockFinder() next, otherwise,
you may want to run champ.runCombat() to eliminat batch effect, then
rerun champ.SVD() to check corrected result.]
```

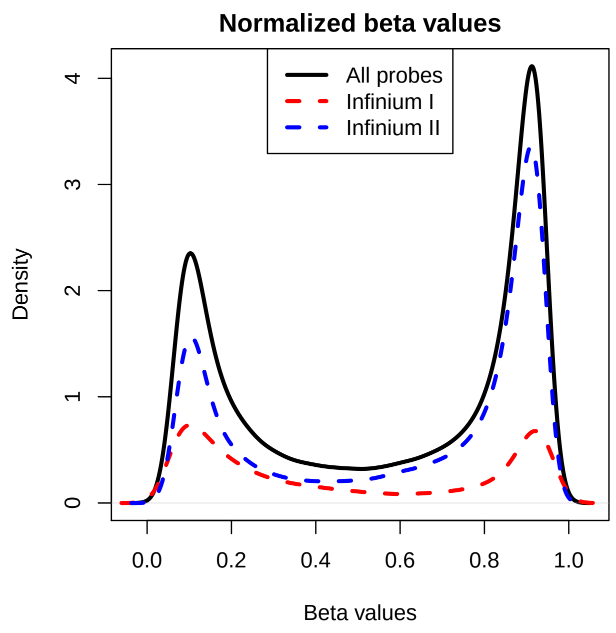


```
champ.SVD(beta=beta.norm.auto,  pd=pData(gset.norm),
          resultsDir=paste(getwd(), "resultsChamp1", sep="/"))

## [===========================]
## [<<< ChAMP.SVD START >>>]
## ---------------
## champ.SVD Results will be saved in /home/murphjes/BIOS6660/Homework_11/resultsChamp1
.
## [SVD analysis will be proceed with 473149 probes and 20 samples.]
```

```
##
## [ champ.SVD() will only check the dimensions between data and pd,
instead if checking if Sample_Names are correctly matched (because
some user may have no Sample_Names in their pd file),thus please make
sure your pd file is in accord with your data sets (beta) and (rgSet).]
## « Following Factors in your pd(sample_sheet.csv) will be analysised:
»
## <Sample_Well>(character):D01, D02, D04, D05, D06, D07, D08, D10,
D11, D12, F01, F02, F04, F05, F06, F07, F08, F10, F11, F12
## <Sample_Plate>(character):Plate 1, Plate 2
## <Array>(character):R01C01, R02C01, R04C01, R05C01, R06C01, R01C02,
R02C02, R04C02, R05C02, R06C02
## <Slide>(character):9721366035, 9992576163
## <Sample_Group>(integer):1, 0
## <child_sex>(factor):M, F
## <xMed>(numeric):13.4171932382483, 14.1502232804958, 13.47522677218,
13.6131563615466, 14.1760952119327, 13.2959841157302, 14.3023176188098,
13.5942079470926, 13.4799696797084, 14.2112799454771, 13.004922678569,
13.6373041221062, 13.313520782848, 13.3162107940661, 12.831109275357,
13.0134971542917, 13.5668270322379, 13.0176783665077, 13.0706241224303,
12.9999119395079
## <yMed>(numeric):13.7270655407971, 10.75359051341, 13.8279354058265,
13.9817453700246, 11.1617550090045, 13.6343004375137, 10.992913326597,
13.9655576614128, 13.829965645917, 10.9173651366522, 13.305624654563,
9.96000193206808, 13.6508801593559, 13.6613670707691, 13.2201801926979,
13.3667216684755, 9.89160384346522, 13.3788336362034, 13.4220647422222,
13.3565832195133
## <predictedSex>(character):M, F
## [champ.SVD have automatically select ALL factors contain at least
two different values from your pd(sample_sheet.csv), if you don't want
to analysis some of them, please remove them manually from your pd
variable then retry champ.SVD().]
##
## « Following Factors in your pd(sample_sheet.csv) will not be analysis:
»
## <Sample_Name>
## <Sample.Group>
## <Pool_ID>
## <Basename>
## <filenames>
## [Factors are ignored because they only indicate Name or Project,
or they contain ONLY ONE value across all Samples.]
##
## « PhenoTypes.lv generated successfully.  »
## « Calculate SVD matrix successfully.  »
## « Plot SVD matrix successfully.  »
## [««« ChAMP.SVD END »»»]
## [===========================]
## [If the batch effect is not significant, you may want to process
champ.DMP() or champ.DMR() or champ.BlockFinder() next, otherwise,
you may want to run champ.runCombat() to eliminat batch effect, then
rerun champ.SVD() to check corrected result.]
```



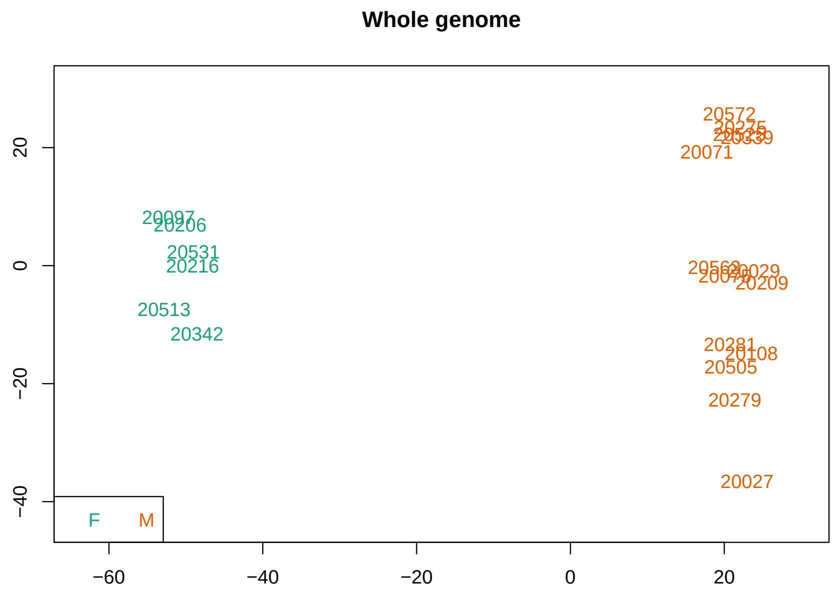
```
#Plot betas after normalization
probe_types = data.frame(cbind(rownames(gset.norm), getProbeType(gset.norm)))
names(probe_types) <- c("Name", "Type")

#Plot density of beta value using average beta
plotBetasByType(rowMeans(beta.norm), probeTypes=probe_types,
                legendPos="top", colors=c("black", "red", "blue"),
                main="Normalized beta values", lwd=3, cex.legend=1)
```



9 Check sex

```
addSex(gset.norm)

## Warning in .pDataAdd(object, sex):  replacing the following columns
in colData(object):  xMed, yMed, predictedSex

## class: GenomicRatioSet
## dim: 484561 20
## metadata(0):
## assays(2): M CN
## rownames(484561): cg13869341 cg14008030 ... cg21106100 cg08265308
## rowData names(6): Probe_rs Probe_maf ... SBE_rs SBE_maf
## colnames(20): 20209 20216 ... 20075 20071
## colData names(14): Sample_Name Sample_Well ... yMed predictedSex
## Annotation
##   array: IlluminaHumanMethylation450k
##   annotation: ilmn12.hg19
## Preprocessing
##   Method: Raw (no normalization or bg correction)
##   minfi version: 1.28.3
##   Manifest version: 0.4.0
```

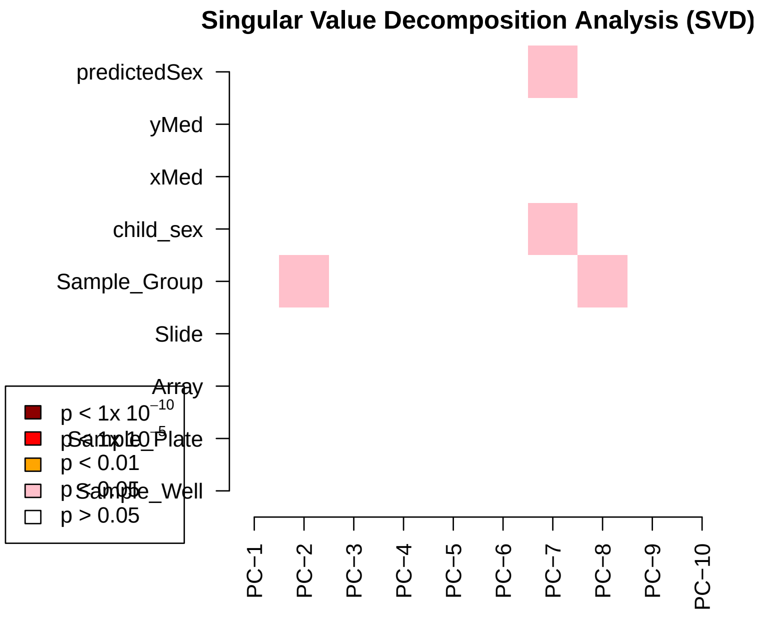
```r
#Identify samples whose clincal sex is different from the predicted sex
table(pData(gset.norm)$child_sex, pData(gset.norm)$predictedSex)

##
##      F  M
##   F  6  0
##   M  0 14

wrongsex = pData(gset.norm)[pData(gset.norm)$predictedSex !=
                              pData(gset.norm)$child_sex, "Sample_Name"]
wrongsex

## integer(0)

mdsPlot(M.norm, numPositions=1000, sampGroups=pData(gset.norm)$child_sex,
        sampNames=pData(gset.norm)$Sample_Name, main="Whole genome")
```



10 Batch correction

```r
pd.norm = pData(gset.norm)

batch = pd.norm$Slide
```

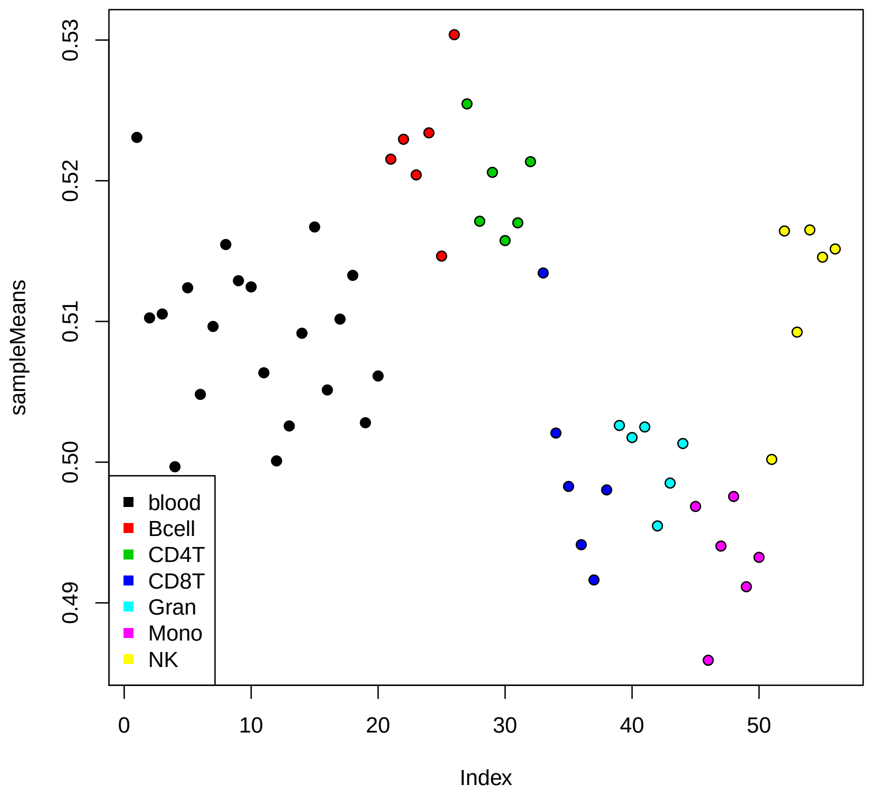
```
#Sample_Group is the exposure satus, which is our main interest
mod = model.matrix(~as.factor(Sample_Group), data=pd.norm)

#M from quantile normalization
M.norm.batch.tmp = ComBat(M.norm, batch, mod, par.prior=T, prior.plots=F)

## Found2batches
## Adjusting for1covariate(s) or covariate level(s)

## Standardizing Data across genes

## Fitting L/S model and finding priors
## Finding parametric adjustments
## Adjusting the Data

nrow(M.norm.batch.tmp)

## [1] 484561

colnames(M.norm.batch.tmp) = sampleNames(mset)

champ.SVD(beta=M.norm.batch.tmp[row.names(M.norm.batch.tmp) %in% row.names(autosomes), ],
          pd=pData(gset.norm), resultsDir=paste(getwd(), "resultsChamp1", sep="/"))

## [===========================]
## [«<< ChAMP.SVD START »»>]
## ----------------
## champ.SVD Results will be saved in /home/murphjes/BIOS6660/Homework_11/resultsChamp1
.
## [SVD analysis will be proceed with 473149 probes and 20 samples.]
##
## [ champ.SVD() will only check the dimensions between data and pd,
instead if checking if Sample_Names are correctly matched (because
some user may have no Sample_Names in their pd file),thus please make
sure your pd file is in accord with your data sets (beta) and (rgSet).]
## « Following Factors in your pd(sample_sheet.csv) will be analysised:
»
## <Sample_Well>(character):D01, D02, D04, D05, D06, D07, D08, D10,
D11, D12, F01, F02, F04, F05, F06, F07, F08, F10, F11, F12
## <Sample_Plate>(character):Plate 1, Plate 2
## <Array>(character):R01C01, R02C01, R04C01, R05C01, R06C01, R01C02,
R02C02, R04C02, R05C02, R06C02
## <Slide>(character):9721366035, 9992576163
## <Sample_Group>(integer):1, 0
## <child_sex>(factor):M, F
## <xMed>(numeric):13.4171932382483, 14.1502232804958, 13.47522677218,
13.6131563615466, 14.1760952119327, 13.2959841157302, 14.3023176188098,
13.5942079470926, 13.4799696797084, 14.2112799454771, 13.004922678569,
13.6373041221062, 13.313520782848, 13.3162107940661, 12.831109275357,
13.0134971542917, 13.5668270322379, 13.0176783665077, 13.0706241224303,
12.9999119395079
## <yMed>(numeric):13.7270655407971, 10.75359051341, 13.8279354058265,
13.9817453700246, 11.1617550090045, 13.6343004375137, 10.992913326597,
13.9655576614128, 13.829965645917, 10.9173651366522, 13.305624654563,
9.96000193206808, 13.6508801593559, 13.6613670707691, 13.2201801926979,
13.3667216684755, 9.89160384346522, 13.3788336362034, 13.4220647422222,
13.3565832195133
## <predictedSex>(character):M, F
## [champ.SVD have automatically select ALL factors contain at least
two different values from your pd(sample_sheet.csv), if you don't want
to analysis some of them, please remove them manually from your pd
variable then retry champ.SVD().]
##
## « Following Factors in your pd(sample_sheet.csv) will not be analysis:
»
## <Sample_Name>
## <Sample.Group>
## <Pool_ID>
## <Basename>
## <filenames>
## [Factors are ignored because they only indicate Name or Project,
or they contain ONLY ONE value across all Samples.]
##
## « PhenoTypes.lv generated successfully.  »
## « Calculate SVD matrix successfully.  »
## « Plot SVD matrix successfully.  »
## [<<<< ChAMP.SVD END >>>>]
## [===========================]
## [If the batch effect is not significant, you may want to process
champ.DMP() or champ.DMR() or champ.BlockFinder() next, otherwise,
you may want to run champ.runCombat() to eliminat batch effect, then
rerun champ.SVD() to check corrected result.]
```



```
#Remove probes that have SNPs or are cross-hybridising
M.norm.batch = rmSNPandCH(M.norm.batch.tmp, dist=2, mafcut=0.05)
colnames(M.norm.batch)

##  [1] "20209" "20216" "20279" "20339" "20342" "20108" "20531" "20275"
##  [9] "20281" "20097" "20505" "20513" "20525" "20563" "20572" "20029"
## [17] "20206" "20027" "20075" "20071"

nrow(M.norm.batch)

## [1] 436536

#get batch corrected beta value
beta.norm.batch = 2^M.norm.batch/(1+2^M.norm.batch)
```

11 Cell type deconvolution

```
#Change data types to match data types of the reference panel data
pData(rgSet_d)$Sample_Name = as.character(pData(rgSet_d)$Sample_Name)
pData(rgSet_d)$Slide = as.numeric(pData(rgSet_d)$Slide)

cellcounts = estimateCellCounts(rgSet_d, compositeCellType="Blood",
             cellTypes=c("CD8T","CD4T", "NK","Bcell","Mono","Gran"),
             returnAll= F, meanPlot=T, verbose=T)

## Loading required package:  FlowSorted.Blood.450k
## [estimateCellCounts] Combining user data with reference (flow sorted)
data.
## Warning in DataFrame(sampleNames = c(colnames(rgSet), colnames(referenceRGset)),
:  'stringsAsFactors' is ignored
## [estimateCellCounts] Processing user and reference data together.
## [preprocessQuantile] Mapping to genome.
## [preprocessQuantile] Fixing outliers.
## [preprocessQuantile] Quantile normalizing.
## [estimateCellCounts] Picking probes for composition estimation.
## [estimateCellCounts] Estimating composition.
```

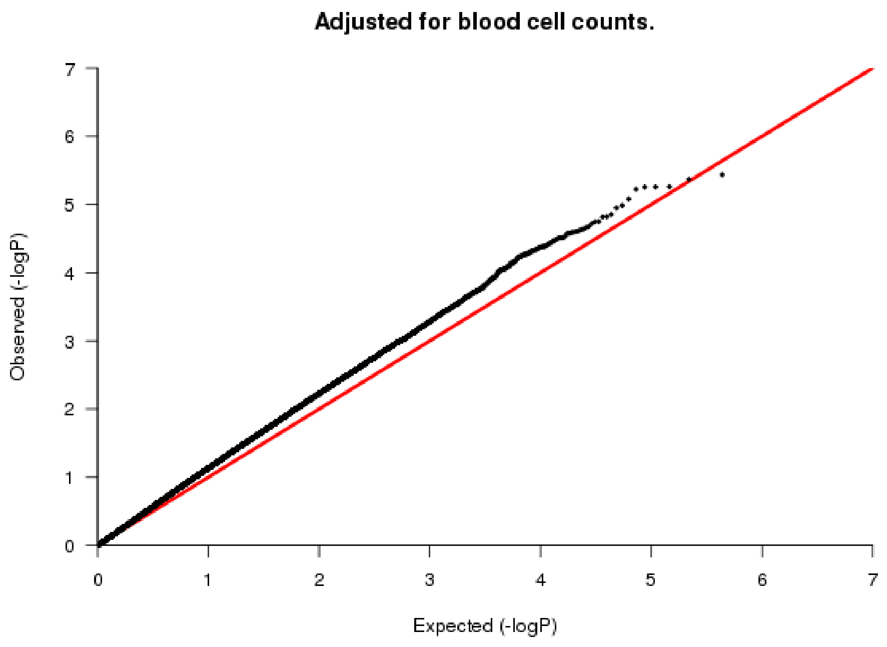


```
rownames(cellcounts)

##  [1] "20209" "20216" "20279" "20339" "20342" "20108" "20531" "20275"
##  [9] "20281" "20097" "20505" "20513" "20525" "20563" "20572" "20029"
## [17] "20206" "20027" "20075" "20071"

colnames(cellcounts)

## [1] "CD8T"  "CD4T"  "NK"    "Bcell" "Mono"  "Gran"

stopifnot(all(rownames(cellcounts)==clindat$Sample_Name))

covariates = cbind(clindat, cellcounts)
```

12 DMP Analysis

```
stopifnot(all(covariates$Sample_Name==colnames(M.norm.batch)))

nCpG = nrow(M.norm.batch)
results1 = data.frame(matrix(NA, nrow=nCpG, ncol=4))
colnames(results1) = c("coef", "se", "pvalue", "adjP")
rownames(results1) = rownames(M.norm.batch)
results2 = data.frame(matrix(NA, nrow=nCpG, ncol=4))
colnames(results2) = c("coef", "se", "pvalue", "adjP")
rownames(results2) = rownames(M.norm.batch)
#CpG = rownames(M.norm.batch)

#this is much faster than looped lm
X = model.matrix(~Exposure + CD8T +
      CD4T + NK + Bcell + Mono + Gran, covariates)
n = nrow(X)
k = ncol(X)-1

betas = t(solve(t(X) %*% X) %*% t(X) %*% t(M.norm.batch))
y_hat = betas %*% t(X)
MSE = rowSums((M.norm.batch-y_hat)^2)/(n-k-1)
sebetas = sqrt(MSE%*%t(diag(solve(t(X)%*%X))))
P = 2*(1-pt(abs(betas/sebetas), n-k-1))

results2[, "coef"] = betas[,"Exposure"]
results2[, "se"] = sebetas[,"Exposure"]
results2[, "pvalue"] = P[,"Exposure"]
```

```r
results2[,'adjP'] = p.adjust(results2[,'pvalue'], method="fdr")
min(results2[,'pvalue'])

## [1] 3.660893e-06

m = ceiling(abs(log10(min(results2[,'pvalue'])))) + 1

sum(results2[,'adjP'] < 0.01)

## [1] 0

sum(results2[,'adjP'] < 0.05)

## [1] 0

sum(results2[,'adjP'] < 0.1)

## [1] 0

sum(results2[,'adjP'] < 0.2)

## [1] 0

observed2 = sort(results2[, "pvalue"])
lobs2 = -log10(observed2)

expected2 = c(1:length(observed2))
lexp2 = -(log10(expected2 / (length(expected2)+1)))
main = "Adjusted for blood cell counts."
plot(c(0,m), c(0,m), col="red", lwd=3, type="l", xlab="Expected (-logP)",
     ylab="Observed (-logP)", xlim=c(0,m), ylim=c(0,m),
     las=1, xaxs="i", yaxs="i", bty="l", main=main)
points(lexp2, lobs2, pch=23, cex=.4, bg="black")
```



```
inflate2 = qchisq(median(results2[,"pvalue"]), df=1, lower.tail = F)/
  qchisq(0.5, df=1, lower.tail = F)
inflate2

## [1] 1.211711

results2[12,]

##                  coef        se    pvalue     adjP
## cg16619049 -0.2491342 0.1768311 0.1842517 0.811931

#Use lm to check the results
y = M.norm.batch[12,]
fit = lm(y ~ as.factor(Exposure)  + CD8T +
    CD4T + NK + Bcell + Mono + Gran, data=covariates)
summary(fit)

##
## Call:
## lm(formula = y ~ as.factor(Exposure) + CD8T + CD4T + NK + Bcell +
##     Mono + Gran, data = covariates)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.43961 -0.20824 -0.07527  0.11613  0.86478
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)           13.0062    11.4250   1.138    0.277
## as.factor(Exposure)1  -0.2491     0.1768  -1.409    0.184
## CD8T                  -9.6046    10.1765  -0.944    0.364
## CD4T                 -13.4197    12.1176  -1.107    0.290
## NK                   -11.5057    11.9242  -0.965    0.354
## Bcell                 -4.8090    13.1094  -0.367    0.720
## Mono                 -13.6598    11.9114  -1.147    0.274
## Gran                 -12.7147    11.3008  -1.125    0.283
##
## Residual standard error: 0.3904 on 12 degrees of freedom
## Multiple R-squared:  0.4464,Adjusted R-squared:  0.1235
## F-statistic: 1.382 on 7 and 12 DF,  p-value: 0.2966
```

```
if(0){
  for (i in 1:nCpG){
    y <- M.norm.batch[i,]
    fit <- lm(y ~ as.factor(Exposure)  + CD8T +
    CD4T + NK + Bcell + Mono + Gran, data=covariates)
    if (substr(rownames(summary(fit)$coefficients)[2], 11, 18) == "Exposure"){
      results1[i, "coef"] <- summary(fit)$coefficients[2,1]
      results1[i, "se"] <- summary(fit)$coefficients[2,2]
      results1[i, "pvalue"] <- summary(fit)$coefficients[2,4]
    }
    else{
      cat(rownames(M.norm.batch)[i], "wrong coefficients\n")
    }
  }

inflate1 = qchisq(median(results1[,"pvalue"]), df=1, lower.tail = F)/
  qchisq(0.5, df=1, lower.tail = F)
inflate1

results1[,'adjP'] = p.adjust(results1[,'pvalue'], method="fdr")

m = ceiling(abs(log10(min(results1[,'pvalue'])))) + 1

sum(results1[,'adjP'] < 0.01)
sum(results1[,'adjP'] < 0.05)
sum(results1[,'adjP'] < 0.1)
sum(results1[,'adjP'] < 0.2)

observed1 = sort(results1[, "pvalue"])
lobs1 = -log10(observed1)

expected1 = c(1:length(observed1))
lexp1 = -(log10(expected1 / (length(expected1)+1)))
main = "Adjusted for blood cell counts."
plot(c(0,m), c(0,m), col="red", lwd=3, type="l", xlab="Expected (-logP)",
       ylab="Observed (-logP)", xlim=c(0,m), ylim=c(0,m),
       las=1, xaxs="i", yaxs="i", bty="l", main=main)
points(lexp1, lobs1, pch=23, cex=.4, bg="black")
}
```

13 DMR Analysis

```
design = model.matrix(~Exposure  + CD8T +
  CD4T + NK + Bcell + Mono + Gran , data = covariates)

colnames(design) = c("(Intercept)", "Exposure",  "CD8T", "CD4T",
                      "NK", "Bcell", "Mono", "Gran")

myannotation = cpg.annotate(datatype = c("array"), object=M.norm.batch,
    arraytype="450K", what="M", analysis.type="differential", coef=2,
    fdr=0.2, design=design)

## Your contrast returned no individually significant probes.  Try
increasing the fdr.  Alternatively, set pcutoff manually in dmrcate()
to return DMRs, but be warned there is an increased risk of Type I
errors.

dmrcoutput = dmrcate(myannotation, lambda=1000, c=2, p.adjust.method = "BH",
                      pcutoff = 0.05, consec = FALSE)

## Fitting chr1...
## Fitting chr10...
## Fitting chr11...
## Fitting chr12...
## Fitting chr13...
## Fitting chr14...
## Fitting chr15...
## Fitting chr16...
## Fitting chr17...
## Fitting chr18...
## Fitting chr19...
## Fitting chr2...
## Fitting chr20...
## Fitting chr21...
## Fitting chr22...
## Fitting chr3...
## Fitting chr4...
## Fitting chr5...
## Fitting chr6...
## Fitting chr7...
## Fitting chr8...
## Fitting chr9...
## Fitting chrX...
## Fitting chrY...
## Demarcating regions...
## Done!
```

```
nrow(dmrcoutput$results)
```

```
## [1] 2251
```

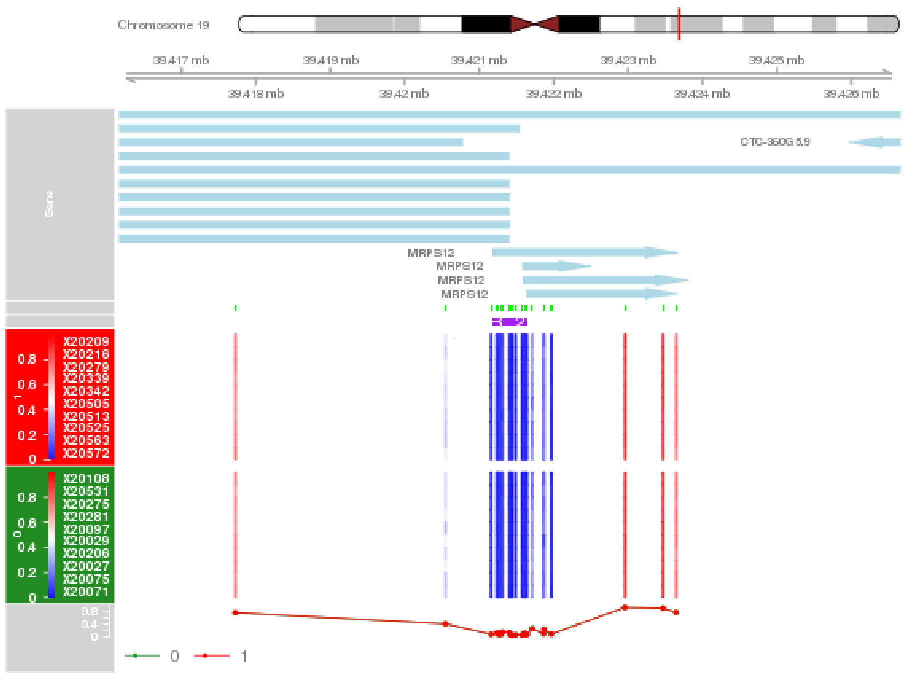
```
dmrcoutput$results[1:5,]
```

```
##                            coord no.cpgs       minfdr  Stouffer   maxbetafc
## 1029 chr12:125287476-125287607       2 1.703146e-07 0.5991892  0.02902030
## 1739   chr17:71739339-71739385       2 3.216073e-05 0.6034509 -0.08012128
## 3754    chr8:74282865-74282931       2 5.340053e-05 0.6067969 -0.17017234
## 2375   chr20:20257861-20258001       2 2.129396e-04 0.6077276 -0.08214293
## 1145 chr13:112669393-112669546       2 5.312888e-05 0.6145133  0.05942925
##       meanbetafc
## 1029  0.02244201
## 1739 -0.06144027
## 3754 -0.13708369
## 2375 -0.08189900
## 1145  0.02096654
```

```
results.ranges = extractRanges(dmrcoutput, genome = "hg19")
results.ranges
```

```
## GRanges object with 2251 ranges and 6 metadata columns:
##        seqnames              ranges strand |   no.cpgs
##           <Rle>           <IRanges>  <Rle> | <integer>
##   1029    chr12 125287476-125287607      * |         2
##   1739    chr17   71739339-71739385      * |         2
##   3754     chr8   74282865-74282931      * |         2
##   2375    chr20   20257861-20258001      * |         2
##   1145    chr13 112669393-112669546      * |         2
##    ...      ...                 ...    ... .       ...
##   3204     chr6   31632171-31633492      * |        39
##   3176     chr6   28889357-28892079      * |        62
##   3222     chr6   32811752-32814322      * |        49
##   3228     chr6   33238093-33240471      * |        46
##   3232     chr6   33266652-33267886      * |        46
##                      minfdr          Stouffer           maxbetafc
##                   <numeric>         <numeric>           <numeric>
##   1029  1.7031462797191e-07 0.599189225247759  0.0290202994953967
##   1739 3.21607273305456e-05 0.603450857514016 -0.0801212780078652
##   3754 5.34005257605472e-05 0.606796912371132   -0.17017234051375
##   2375 0.000212939638209635 0.607727574577886 -0.0821429279590144
##   1145  5.3128879890256e-05 0.614513304573859  0.0594292526455031
##    ...                  ...               ...                 ...
##   3204  0.00309106786906072                 1  0.0534397741112885
##   3176 7.07089362002807e-05                 1 -0.0262245481401839
##   3222 7.52517722584033e-08                 1 -0.0444864086796222
##   3228  0.00170281642555064                 1 -0.0448684178278327
##   3232  0.00699002120409941                 1 -0.0494220880317316
##                   meanbetafc
##                    <numeric>
##   1029     0.022442011250183
##   1739   -0.0614402707632263
##   3754     -0.13708368825764
##   2375   -0.0818989979116109
##   1145     0.020966538773313
##    ...                   ...
##   3204   0.00279058683028906
##   3176  0.000965370814145147
##   3222  -0.00149543173577999
##   3228  -0.00159397626667104
##   3232 -0.000153449747351513
##
##
##   1029
##   1739
##   3754
##   2375
##   1145
##    ...
##   3204 Y_RNA.248-201, CSNK2B-008, CSNK2B-009, CSNK2B-LY6G5B-1181-001, CSNK2B-001, CSNK2B-
##   3176
##   3222                                                                  PSMB8-001, PSMB8
##   3228
##   3232
##   -------
##   seqinfo: 23 sequences from an unspecified genome; no seqlengths
groups = c("1"="red", "0"="forestgreen")
```

```
cols = groups[as.character(covariates$Exposure)]

DMR.plot(ranges=results.ranges, dmr=2201, CpGs=beta.norm.batch,
         what="Beta", arraytype="450K", phen.col=cols, genome="hg19")
```


Chromosome 19
39.417 mb
39.419 mb
39.421 mb
39.423 mb
39.425 mb
39.418 mb
39.42 mb
39.422 mb
39.424 mb
39.426 mb
Gene
CTC-360G5.9
MRPS12
MRPS12
MRPS12
MRPS12
X20209
X20216
X20279
X20339
X20342
X20505
X20513
X20525
X20563
X20572
X20108
X20531
X20275
X20281
X20097
X20029
X20206
X20027
X20075
X20071