BIOS 7659 Homework 8

Tim Vigers

26 November 2020

# 1. Cell-Type Composition

Read in the data:

baseDir1 = "blood/plate1"  
targets1 = read.metharray.sheet(baseDir1)

## [1] "blood/plate1/selected\_plate1.csv"

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

## [1] "blood/plate2/selected\_plate2.csv"

targets = rbind(targets1, targets2)  
rgSet = read.metharray.exp(targets=targets,extended=T)  
sampleNames(rgSet) = rgSet[[1]]  
getManifest(rgSet)

## 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("blood/demographic.txt", sep="\t", header=T)  
pData(rgSet)$Sample\_Group = clindat$Exposure  
pData(rgSet)$child\_sex = clindat$child\_sex

## a) Find differentially methylated positions based on exposure status

# SWAN normalization  
msetSWAN = preprocessSWAN(rgSet)  
dmp = dmpFinder(msetSWAN,pheno = msetSWAN$Sample\_Group,type = "continuous")

Here the unexposed group is the reference, so a negative value of beta in the DMP table indicates that the probe was hypomethylated in the exposed group (and a positive value indicates hypermethylation). There are no probes significant at the level, but there are 4 significant at the level:

dmp %>% rownames\_to\_column(var="Probe") %>% filter(pval < 1e-5) %>% mutate(Direction = ifelse(sign(beta)==-1,"Hypomethylated","Hypermethylated")) %>% select(Probe,beta,Direction,pval) %>% rename("Beta"=beta,"P Value"=pval) %>% flextable(.) %>% footnote(.,i=1,j=3,value = as\_paragraph("With respect to exposure"),ref\_symbols = "\*",part = "header") %>% set\_table\_properties(.,layout = "autofit",width = 0.75)

| Probe | Beta | Direction\* | P Value |
| --- | --- | --- | --- |
| cg25761397 | -0.3997037 | Hypomethylated | 1.332702e-06 |
| cg00244352 | -0.3431909 | Hypomethylated | 4.254079e-06 |
| cg15614119 | -0.2460500 | Hypomethylated | 5.528681e-06 |
| cg21088165 | -0.4816779 | Hypomethylated | 7.535189e-06 |
| \*With respect to exposure | | | |

## b) Cell-type composition

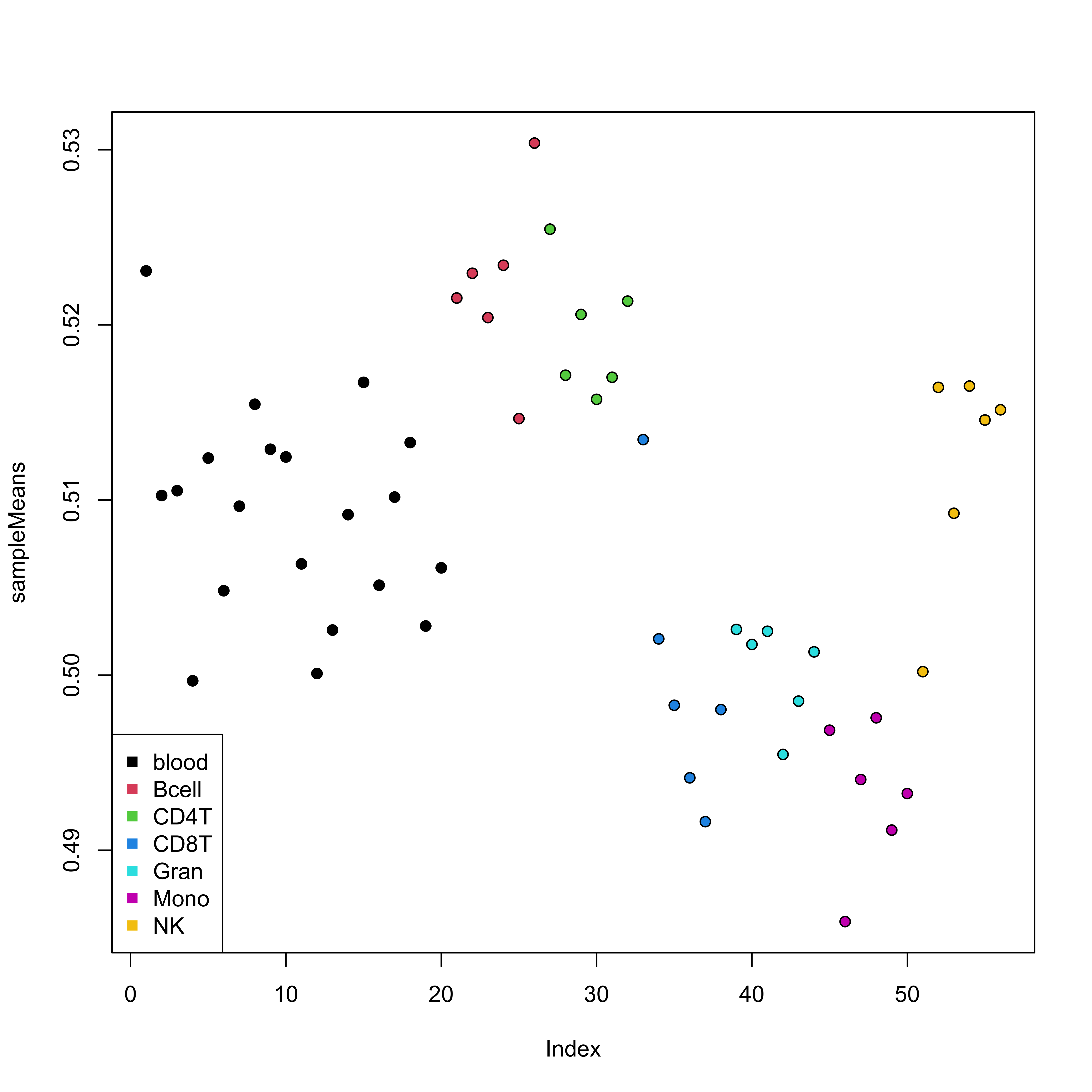
Cell-type composition is important in methylation studies because methylation is a key factor in converting cells from a pluripotent progenitor to specific cell-type. As a result, differentially methylated regions (DMRs) can distinguish cell-types with high sensitivity and specificity [1]. In other words, if you are comparing two samples with different cell-type composition without some sort of adjustment, many DMRs that distinguish between samples will likely be the result of cell-type composition rather than the exposure of interest.

There are many ways to adjust for cell-type composition when cell-type isn’t directly measured, but the approach in Houseman et al. [1] is implemented in estimateCellCounts() in the minfi package, and that paper is frequently cited. The approach in Jaffe & Irizarry [2] is simply an adaptation of the Houseman algorithm for the 450K array. These approaches require reference data, but there are also reference-free methods such as singular value decomposition, linear models with principal components, SVA, RUV, RUVm, and others. However, the best option (if possible) is to measure complete blood counts rather than estimating cell types.

The Houseman algorithm takes an vector of methylation values for a homogenous cell population, and proposes the linear model where is a covariate vector indicating cell-type and is an matrix. Also, indexes the specimen and is an error vector. Next, they propose a similar model for the data without cell-type measurements: with the linking regression model where indexes samples of interest and is a matrix of errors. The math gets fairly complicated after this, so I won’t go into any more detail, but the points is that the cell-type mixture coefficients can be “recovered” from by estimating and via linear models using ordinary least squares, mixed effects models, limma, or surrogate variable analysis. The authors also recommend bootstrap procedures for estimating standard errors.

## c) Estimate cell counts

rgSet = read.metharray.exp(targets=targets,extended=F)  
cells = estimateCellCounts(rgSet,meanPlot = T,compositeCellType = "Blood",cellTypes = c("CD8T","CD4T","NK","Bcell","Mono","Gran"))



cells %>% as.data.frame(.) %>% flextable(.) %>% set\_table\_properties(.,layout = "autofit") %>% set\_caption(.,"Estimated Cell-Type Proportions")

Estimated Cell-Type Proportions

| CD8T | CD4T | NK | Bcell | Mono | Gran |
| --- | --- | --- | --- | --- | --- |
| 0.24473013 | 0.21221126 | 1.179639e-02 | 0.12211486 | 0.13014145 | 0.3087963 |
| 0.07834664 | 0.18399994 | 7.538122e-02 | 0.08690504 | 0.09264915 | 0.4918255 |
| 0.14571184 | 0.28648393 | 3.440698e-02 | 0.10895127 | 0.13164621 | 0.3002824 |
| 0.09831487 | 0.28551019 | 7.911824e-02 | 0.14558753 | 0.11676131 | 0.2611704 |
| 0.19685961 | 0.22271280 | 4.197430e-02 | 0.10295708 | 0.11272311 | 0.3369362 |
| 0.14994330 | 0.17078773 | 1.096162e-01 | 0.06853485 | 0.11621413 | 0.3868448 |
| 0.15172152 | 0.27356727 | 1.285225e-01 | 0.06991721 | 0.05034163 | 0.3301254 |
| 0.22450473 | 0.34554033 | 9.895694e-02 | 0.14560668 | 0.07658245 | 0.1192818 |
| 0.07678494 | 0.20818682 | 4.740060e-02 | 0.10282402 | 0.10651698 | 0.4705302 |
| 0.10358519 | 0.25831494 | 8.881471e-02 | 0.09553022 | 0.08252781 | 0.3840780 |
| 0.10510257 | 0.30098768 | -3.469447e-18 | 0.11750532 | 0.06812916 | 0.4073822 |
| 0.11391563 | 0.21834668 | 2.652012e-01 | 0.08757173 | 0.16612150 | 0.1376624 |
| 0.02934473 | 0.11473780 | 1.011562e-01 | 0.07603434 | 0.10906073 | 0.5655494 |
| 0.02019483 | 0.07077094 | 7.098415e-02 | 0.08893996 | 0.11387411 | 0.6443090 |
| 0.18778138 | 0.14126929 | 1.800243e-01 | 0.07174439 | 0.07516621 | 0.3642867 |
| 0.12504738 | 0.12085375 | 3.051989e-02 | 0.08876738 | 0.10867516 | 0.5284240 |
| 0.18194673 | 0.20379202 | 5.891531e-02 | 0.09304232 | 0.11067444 | 0.3619681 |
| 0.09117649 | 0.11894417 | 1.483735e-01 | 0.11561809 | 0.20884701 | 0.3316215 |
| 0.13787875 | 0.15443145 | 1.035689e-01 | 0.09733477 | 0.11691844 | 0.3874561 |
| 0.12538598 | 0.20456790 | 5.828551e-03 | 0.12964674 | 0.10794735 | 0.4276791 |

The plot above shows the average methylation across cell-type differentiating probes, for our data and the reference data set (“FlowSorted.Blood.450k”). The means of the two datasets should be in a similar range. If not, this suggests large batch effects in the data and the composition estimates should not be trusted (the function automatically performs quantile normalization between the two datasets to reduce potential batch effects). This plot looks reasonable, so we can probably trust these cell-type estimates and repeat part a) with adjustment for cell-type composition.

Because the cell proportions necessarily sum to 1, we technically should not be treating them as independent variables. One option to get around this issue is to simply drop one of the cell types from the model, so for the purposes of this homework I’ve decided not to include the NK cells, because these generally had very small estimated proportions.

# Format data  
m = getM(msetSWAN)  
cells = cbind(msetSWAN$Sample\_Group,cells)  
colnames(cells)[1] = "Sample\_Group"  
cells = cells[,-which(colnames(cells)=="NK")]  
# Fit models with limma  
d = model.matrix(~ Sample\_Group + CD8T + CD4T + Bcell + Mono + Gran,as.data.frame(cells))  
fit = lmFit(m,d)  
sd = fit$stdev.unscaled  
beta = fit$coefficients[,"Sample\_Group"]  
# Calculate p values  
pval = 2\*pt(abs(beta / (sd[,"Sample\_Group"] \* fit$sigma)),  
 df = fit$df.residual,lower.tail = F)  
qval = p.adjust(pval,"fdr")  
dmp = cbind(fit$coefficients,pval)  
dmp = cbind(dmp,qval)

One thing I noticed when fitting these models is that many of the regression estimates for cell type are really large with correspondingly large standard errors. When I removed them from the model and re-ran the code I got the same results as dmpFinder() in part a), so I don’t think it’s a programming issue. These estimates may not be a problem, but with more time I would probably investigate them a little. However, for this homework I am assuming that the estimates are okay given that we aren’t interested in inference for cell type. Also, I did copy the idea of using limma::lmFit() from Karen Kanaster because it is so much faster than lm(), but the code above is my own.

There are 0 probes significant at the level, and 21 probes significant at the level:

dmp %>% as.data.frame(.) %>% rownames\_to\_column(var="Probe") %>% filter(pval < 1e-5) %>% mutate(Direction = ifelse(sign(Sample\_Group)==-1,"Hypomethylated","Hypermethylated")) %>% select(Probe,Sample\_Group,Direction,pval) %>% rename("Beta"=Sample\_Group,"P Value"=pval) %>% flextable(.) %>% footnote(.,i=1,j=3,value = as\_paragraph("With respect to exposure"),ref\_symbols = "\*",part = "header") %>% set\_table\_properties(.,layout = "autofit",width = 0.75)

| Probe | Beta | Direction\* | P Value |
| --- | --- | --- | --- |
| cg09743500 | -0.2911310 | Hypomethylated | 9.750163e-06 |
| cg10104683 | -0.5906237 | Hypomethylated | 2.805849e-06 |
| cg02493905 | -0.3558310 | Hypomethylated | 3.558430e-06 |
| cg08644365 | -0.2323063 | Hypomethylated | 4.600940e-06 |
| cg06759518 | -0.4391959 | Hypomethylated | 1.351683e-06 |
| cg22685502 | -0.3962258 | Hypomethylated | 3.693655e-07 |
| cg11088968 | -0.3627167 | Hypomethylated | 4.468435e-06 |
| cg00850971 | 0.2076540 | Hypermethylated | 2.086687e-06 |
| cg22566787 | -0.4480062 | Hypomethylated | 9.697822e-06 |
| cg23313379 | -0.4556526 | Hypomethylated | 9.528730e-06 |
| cg25761397 | -0.3920133 | Hypomethylated | 9.302411e-06 |
| cg02902261 | -0.2455404 | Hypomethylated | 8.240024e-06 |
| cg08126560 | -0.2854584 | Hypomethylated | 8.318343e-06 |
| cg13912480 | -0.5259333 | Hypomethylated | 3.881068e-06 |
| cg05307477 | -0.3158704 | Hypomethylated | 4.012998e-06 |
| cg06296890 | -0.5336542 | Hypomethylated | 4.391571e-06 |
| cg19258882 | -0.2377388 | Hypomethylated | 8.622982e-06 |
| cg24566687 | 0.3977008 | Hypermethylated | 5.135834e-06 |
| cg15431137 | -0.3232366 | Hypomethylated | 2.896632e-06 |
| cg03605666 | -0.3257203 | Hypomethylated | 4.520154e-06 |
| cg00549412 | -0.4226530 | Hypomethylated | 6.932961e-06 |
| \*With respect to exposure | | | |

After adjusting for cell type composition, there are a lot more significant probes, and the majority of them are still hypomethylated with respect to the exposure. If there were no changes to the results that would suggest that the cell types were pretty even distributed among all the samples, but this suggests that there are important cell-type differences between samples. In addition to cell type, I would also definitely adjust for age and sex. Both of these are known to be associated with methylation patterns, so it’s important to collect that information in all genomics studies and to adjust for them in all analyses.

# 2. ChIP-Seq

Read in the data:

data(yeastCDS)  
allData = dataReading("./chipseq/tup1\_IP.txt", "./chipseq/mock\_IP.txt",yeastSpecies = yeastCDS$Saccharomyces.cerevisiae)

## [1] "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"  
## [1] "Reading IP and control datasets... "  
## [1] "... done"  
## [1] "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"  
## [1] ""

## a) What methods were used?

Sequencing was performed using Illumina HiSeq 2000 or SOLiD V4. Then, cDNA was labeled with Cy3 and hybridized to the NimbleGen S. cerevisiae HX12 array. This array was scanned with a GenePix 4000B, and the image processed using NimbleScan. These signal intensities were normalized to other NimbleGen arrays collected by the Park lab [3].

Next, deep sequencing reads were mapped to the unmasked sacCer3 reference using BWA (with default options), and non-uniquely mapped reads were filtered out. Peak calling was performed using Model-based Analysis of ChIP-Seq (MACS2). For the input control, a Swi6 ChIP sample was sequenced without the immunoprecipitation step. This is “the sheared chromatin from a SWI6 TAP-tagged strain.” [3] On the other hand, the mock sample is from wild-type yeast (with no TAP-tagged protein expression) chromatin, but was incubated with a non-specific immunoglobulin G (IgG) antibody.

## b) Average number of sequences mapped

baseLineIP = baseLineCalc(allData$IPdata[,ncol(allData$IPdata)])  
baseLineControl = baseLineCalc(allData$controlData[,ncol(allData$controlData)])

The average number of sequences mapped in the Tup1 ChIP sample is 65.5266955 and the average number in the mock sample is 14.4172567, so the Tup1 ChIP sample appears to have a lot more reads.

## c) Peak detection

# Subset the data  
ip = allData$IPdata[allData$IPdata$V1 == "chrV",]  
control = allData$controlData[allData$controlData$V1 == "chrV",]  
peaks = peakDetection(ip$V3,control$V3,"chrV",baseLineIP = baseLineIP,baseLineControl = baseLineControl,outputName = "bPeaks\_results\_mock")

## [1] "1% of windows were analyzed"  
## [1] "5% of windows were analyzed"  
## [1] "10% of windows were analyzed"  
## [1] "20% of windows were analyzed"  
## [1] "50% of windows were analyzed"  
## [1] "60% of windows were analyzed"  
## [1] "70% of windows were analyzed"  
## [1] "80% of windows were analyzed"  
## [1] "90% of windows were analyzed"  
## [1] "100% of windows were analyzed"  
## [1] "66 significant window(s) were detected..."  
## [1] "... starting merging procedure"  
## [1] ""  
## [1] "# of detected basic peaks (bPeaks) : 17"  
## [1] ""  
## [1] "\*\* Saving chromosome information in PDF file:"  
## [1] "bPeaks\_results\_mock\_dataSummary.pdf"  
## [1] ""

## [1] "\*\* Bed file saving in:"  
## [1] "bPeaks\_results\_mock.bed"  
## [1] ""  
## [1] "\*\* Peak drawing in PDF file:"  
## [1] "bPeaks\_results\_mock\_bPeaksDrawing.pdf"

The peakDetection() function found 17 peaks. Peaks are detected using a sliding window along the genome (specified by the windowSize parameter) and based on two global parameters and four local parameters. The global parameters and are calculated using the baseLineCalc() function above, and represent the sequencing coverage in IP and control samples across the entire genome (e.g.  for the input sample). The local parameters for each window are:

where represents the number of nucleotides in window and and represent the number of sequences mapped at position in input and control samples, respectively. Positive windows are selected if . The arguments IPthreshold, controlThreshold, ratioThreshold, averageThreshold allow the user to specify , , , and respectively. Multiple positive windows are then merged into single peaks.

![bPeaks Data Summary for Mock Sample](data:application/pdf;base64,)

bPeaks Data Summary for Mock Sample

In the PDF produced by bPeaks (above), the parameter and results summaries do what they say on the tin and summarize the parameters and results. The bottom left figure plots and for each window, and the red lines indicate the chosen thresholds (in this case the defaults). Positive windows must have IP signal > and control signal < , and windows that meet these requirements are highlighted in red. The bottom right figure plots and . Positive windows must have logFC > and density of reads > , so the windows meeting these requirements are again highlighted in red.

## d)

There are many good alternatives to bPeaks that use some sort of statistical test, including Useq (binomial), MACS, PeakSeq, CSAR (Poisson), CisGenome (negative binomial), and ZINBA (zero-inflated negative binomial). Hpeak and BayesPeak use hidden Markov models. I don’t have time to describe all of them, but given these choices I would probably use MACS because it seems to be commonly used, and the documentation looks clear and in-depth.

First, MACS (which stands for **M**odel-based **A**nalysis of **C**hIP-**S**eq) linearly scales the control read count to match the total input count, then removes duplicate reads. Then, read distribution along the genome is modeled with a Poisson distribution and the parameter is estimated. Similar to bPeaks, MACS then slides a window along the genome to find candidate peaks based on Poisson() distribution p-values < . Then for each candidate peaks, MACS finds where and are estimated from the 5 kb or 10 kb window centered at the candidate peak location. allows MACS to calculate a p value for each candidate peak, rather than simply relying on fold change.[4]

## e) Peak location

# Find peaks  
peak\_location = peakLocation("./bPeaks\_results\_mock.bed",yeastCDS$Saccharomyces.cerevisiae,outputName = "bPeaksLocation\_mock")

## [1] "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"  
## [1] "Opening BED file with peak information:"  
## [1] "./bPeaks\_results\_mock.bed"  
## [1] "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"  
## [1] ""  
## [1] "Starting peak location regarding ORF/CDS positions..."  
## [1] ""  
## [1] "# of analyzed peaks: 17"  
## [1] "# of peaks UPSTREAM annotated CDS : 15"  
## [1] "# of peaks IN annotated CDS : 2"  
## [1] ""

## [1] "Saving the results in:"  
## [1] "bPeaksLocation\_mock\_peakLocation\_inPromoters.txt"  
## [1] "bPeaksLocation\_mock\_peakLocation\_inCDS.txt"  
## [1] "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

# Read results back in  
promoters = read.delim("./bPeaksLocation\_mock\_peakLocation\_inPromoters.txt",header = F)  
cds = read.delim("./bPeaksLocation\_mock\_peakLocation\_inCDS.txt",header = F)

Out of the 17 peaks detected, 15 are in promoters and 2 are in CDS (genes).

## f) Repeat using the input IP sample

# Read in data  
allData = dataReading("./chipseq/tup1\_IP.txt","./chipseq/input\_IP.txt",yeastSpecies = yeastCDS$Saccharomyces.cerevisiae)

## [1] "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"  
## [1] "Reading IP and control datasets... "  
## [1] "... done"  
## [1] "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"  
## [1] ""

# Baseline  
baseLineControl = baseLineCalc(allData$controlData[,ncol(allData$controlData)])  
# Peak detection  
control = allData$controlData[allData$controlData$V1 == "chrV",]  
peaks = peakDetection(ip$V3,control$V3,"chrV",baseLineIP = baseLineIP,baseLineControl = baseLineControl,outputName = "bPeaks\_results\_input")

## [1] "1% of windows were analyzed"  
## [1] "5% of windows were analyzed"  
## [1] "10% of windows were analyzed"  
## [1] "20% of windows were analyzed"  
## [1] "50% of windows were analyzed"  
## [1] "60% of windows were analyzed"  
## [1] "70% of windows were analyzed"  
## [1] "80% of windows were analyzed"  
## [1] "90% of windows were analyzed"  
## [1] "100% of windows were analyzed"  
## [1] "63 significant window(s) were detected..."  
## [1] "... starting merging procedure"  
## [1] ""  
## [1] "# of detected basic peaks (bPeaks) : 19"  
## [1] ""  
## [1] "\*\* Saving chromosome information in PDF file:"  
## [1] "bPeaks\_results\_input\_dataSummary.pdf"  
## [1] ""

## [1] "\*\* Bed file saving in:"  
## [1] "bPeaks\_results\_input.bed"  
## [1] ""  
## [1] "\*\* Peak drawing in PDF file:"  
## [1] "bPeaks\_results\_input\_bPeaksDrawing.pdf"

# Peak locations  
peak\_location = peakLocation("./bPeaks\_results.bed",yeastCDS$Saccharomyces.cerevisiae,outputName = "bPeaksLocation\_input")

## [1] "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"  
## [1] "Opening BED file with peak information:"  
## [1] "./bPeaks\_results.bed"  
## [1] "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"  
## [1] ""  
## [1] "Starting peak location regarding ORF/CDS positions..."  
## [1] ""  
## [1] "# of analyzed peaks: 19"  
## [1] "# of peaks UPSTREAM annotated CDS : 18"  
## [1] "# of peaks IN annotated CDS : 2"  
## [1] ""

## [1] "Saving the results in:"  
## [1] "bPeaksLocation\_input\_peakLocation\_inPromoters.txt"  
## [1] "bPeaksLocation\_input\_peakLocation\_inCDS.txt"  
## [1] "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"

promoters = read.delim("./bPeaksLocation\_input\_peakLocation\_inPromoters.txt",header = F)  
cds = read.delim("./bPeaksLocation\_input\_peakLocation\_inCDS.txt",header = F)

The average number of sequences mapped in the input IP sample is 63.733644, which is much closer the Tup 1 sample (65.5266955) than the mock sample was. With the new control data, the peakDetection() function found 19 peaks. Of these peaks, 18 are in promoters and 2 are in genes, so at least 1 peak must be in both a gene and promoter.

The Park et al. [3] paper found that expression bias due to genes that are transcribed at high rates was not effectively controlled for by the input sample. So, they recommend using “mock ChIP data to minimize false positives.” Our results here generally agree because we found two positives when using the input control that were eliminated when using the mock sample. Also, in the data summary for the input sample the cutoff is so large that it isn’t even visible on the plot. This is a good visual representation of the false positives, because there are several windows to the right side of the x axis that would likely be excluded with a better control sample. However, the differences between the two do not seem drastic to me, so I think it would be okay to use the input sample if it’s the only option available.

# References

1. Houseman EA, Accomando WP, Koestler DC, et al. DNA methylation arrays as surrogate measures of cell mixture distribution. BMC Bioinformatics. 2012;13(1):86. <doi:10.1186/1471-2105-13-86>
2. Jaffe AE, Irizarry RA. Accounting for cellular heterogeneity is critical in epigenome-wide association studies. Genome Biol. 2014;15(2):R31. <doi:10.1186/gb-2014-15-2-r31>
3. Park D, Lee Y, Bhupindersingh G, Iyer VR. Widespread misinterpretable ChIP-seq bias in yeast. PloS One. 2013;8(12):e83506. <doi:10.1371/journal.pone.0083506>
4. Zhang Y, Liu T, Meyer CA, et al. Model-based Analysis of ChIP-Seq (MACS). Genome Biol. 2008;9(9):R137. <doi:10.1186/gb-2008-9-9-r137>