GRHL2

Andrew Holding

1/18/2018

Pre-processing

Download files

```
#!/bin/sh
java -jar ../java/clarity-tools.jar -l SLX-14333
```

Removal of reads in blacklisted sites

```
#downloaded from http://mitra.stanford.edu/kundaje/akundaje/release/blacklists/hg38-human/hg38.blacklis

bl=../blacklists/hg38.blacklist.bed


mkdir ./blacklist_filtered
cd SLX-14333
for f in *.bam
do
    echo $f
    bedtools intersect -v -abam $f -b $bl  > ../blacklist_filtered/$f
done

### Re-index

cd ../blacklist_filtered
for f in *.bam
do
samtools index $f
done
```

Peak Calling

```
### MACS peak caller

### Run macs on the blacklisted data
mkdir ./peaks
cd peaks
control=../GRHL2_filtered/SLX-14333.D701_D503.bam
for bam in ../GRHL2_filtered/*.bam
do
root=`basename $bam .bam`
macs2 callpeak -t $bam -c $control -f BAM -n $root -g hs &
done
```

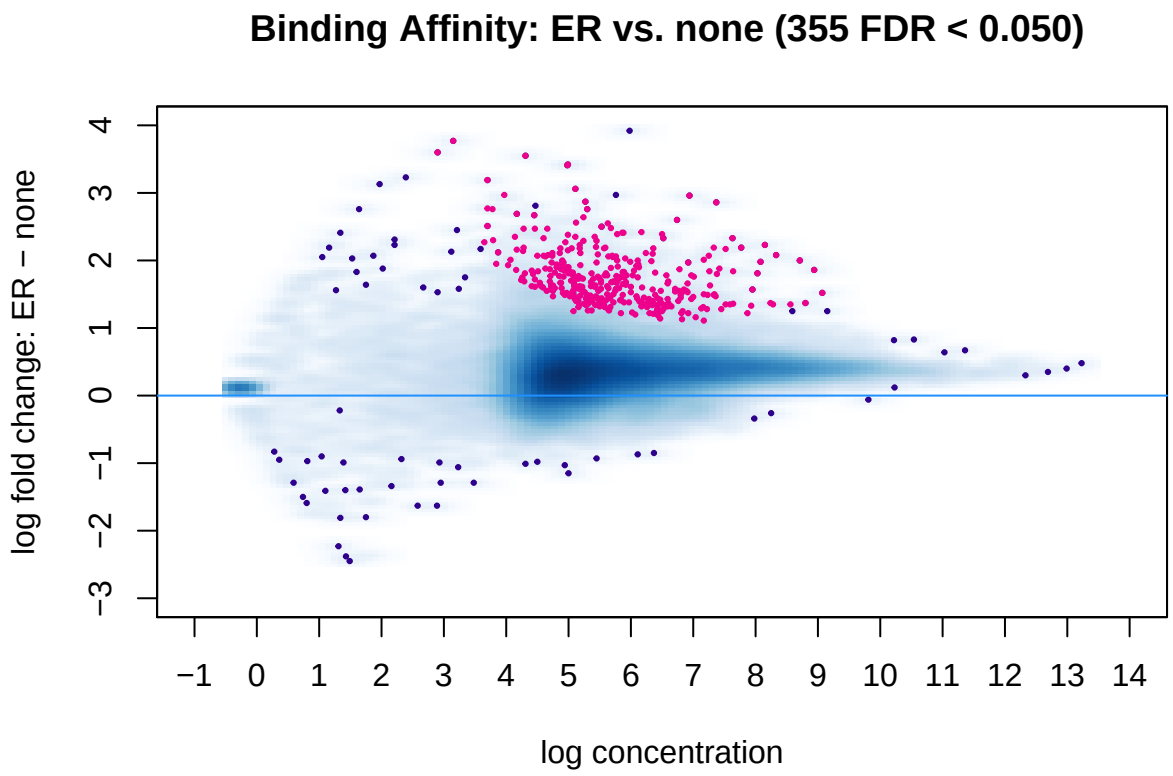


Figure 1: MA plot showing changes in GRHL2 binding before and after treatment with 100nM E2.

Differntial binding analysis

MA plot

```
suppressMessages(library(DiffBind))

if(!file.exists("rdata/003_diffbind.rda")) {
    GRHL2 <- dba(sampleSheet="samplesheet/samplesheet.csv")
    GRHL2 <- dba.count(GRHL2, summits=250)
    GRHL2 <- dba.contrast(GRHL2)
    GRHL2 <- dba.analyze(GRHL2)
    save(GRHL2,file="rdata/003_diffbind.rda")
} else {
    load("rdata/003_diffbind.rda")
}


dba.plotMA(GRHL2, bFlip=1)
```

PCA

```
dba.plotPCA(GRHL2,components = 2:3)
```

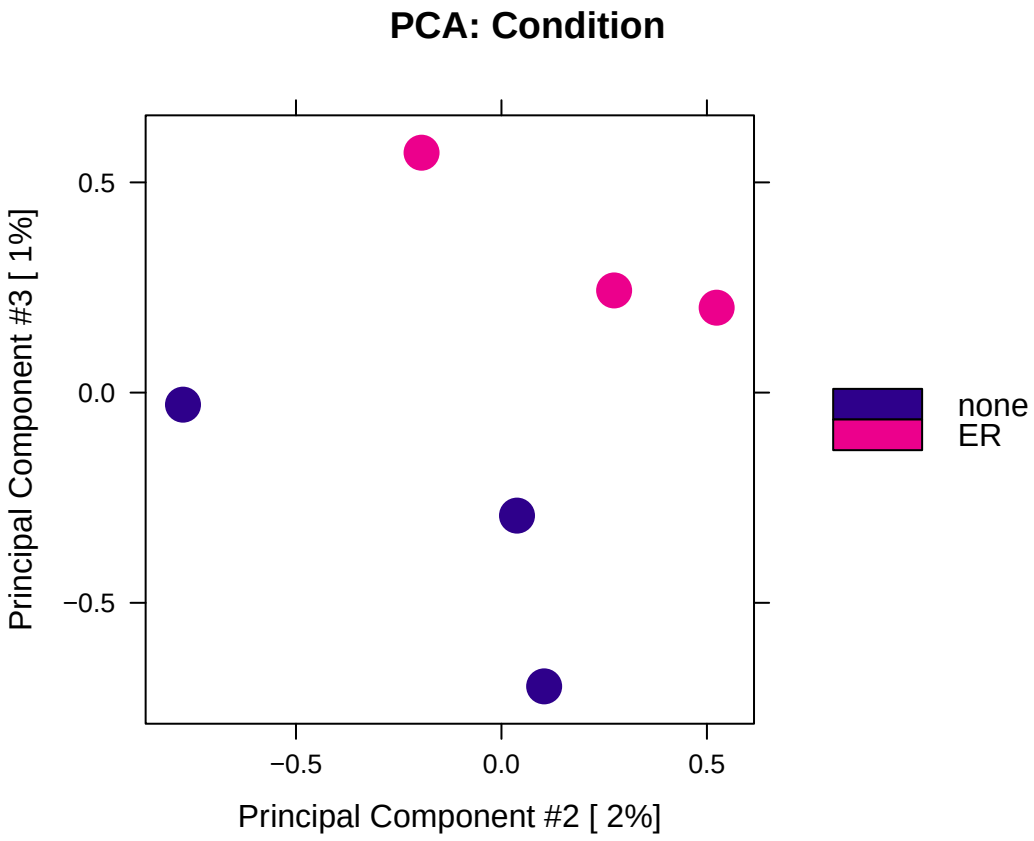


Figure 2: PCA Plot showing clustering of samples by condition

Table of differntiall bound sites

```
library(knitr)
kable(head(as.data.frame(dba.report(GRHL2))))
```

	seqnames	start	end	width	strand	Conc	Conc_none	Conc_ER	Fold	p.value	FDR
12123	14	93959394	93959894	501	*	7.63	6.04	8.37	-2.33	0	0e+00
9047	12	75706270	75706770	501	*	6.74	4.93	7.52	-2.60	0	0e+00
24493	21	31529399	31529899	501	*	6.50	4.86	7.25	-2.39	0	0e+00
129	1	7447769	7448269	501	*	4.98	2.44	5.85	-3.41	0	0e+00
40054	9	75101020	75101520	501	*	6.94	4.80	7.76	-2.96	0	0e+00
28564	3	176953741	176954241	501	*	5.53	3.80	6.30	-2.50	0	1e-07

Number of sites with increased or decreased binding

```
r<-dba.report(GRHL2,th=1)
length(r[r$Fold>0])
```

```
## [1] 4973
```

```
length(r[r$Fold<0])
```

```
## [1] 37496
```

Quality Control

Reproducability of peaks

```
dba.plotVenn(GRHL2,GRHL2$masks$ER, label1="Rep1", label2="Rep2", main="GRHL2 +E2")
```

```
dba.plotVenn(GRHL2,GRHL2$masks$none, label1="Rep1", label2="Rep2", main="GRHL2 -E2")
```

```
called_none<-rowSums(GRHL2$called[,c(1:3)])
called_ER<-rowSums(GRHL2$called[,c(4:6)])
print(paste("Peaks called in -E2 samples:", length(called_none[called_none>0])))
```

```
## [1] "Peaks called in -E2 samples: 38763"
```

```
print(paste("Peaks called in +E2 samples:", length(called_ER[called_ER>0])))
```

```
## [1] "Peaks called in +E2 samples: 42565"
```

```
called_both<-called_none*called_none
print(paste("Peaks called in both settings:", length(called_both[called_both>0])))
```

```
## [1] "Peaks called in both settings: 38763"
```

VULCAN Analysis

```
suppressMessages(library(vulcan))
```

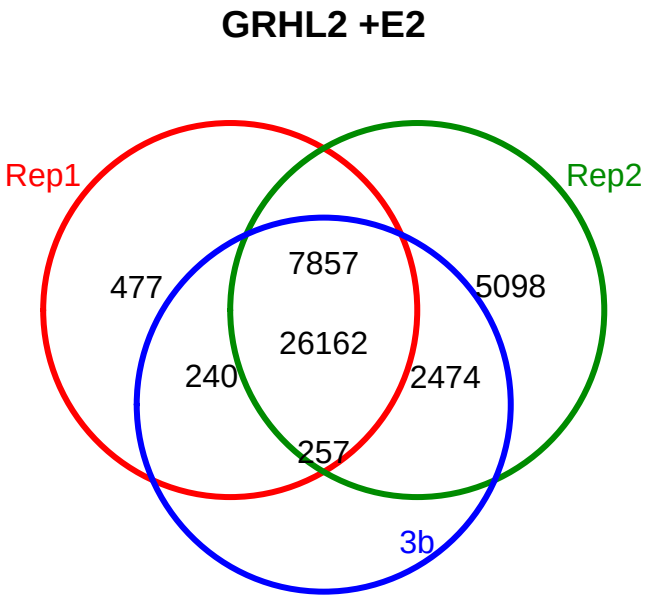


Figure 3: Peak overlap +E2

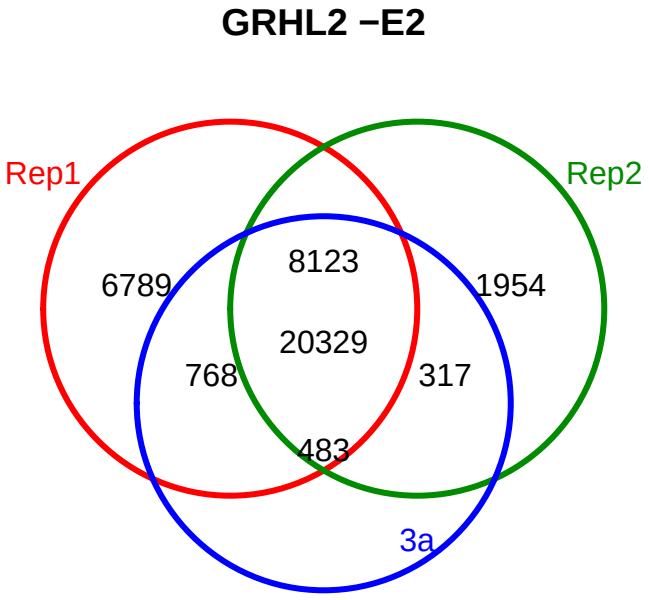


Figure 4: Peak overlap -E2

```
dist_calc<-function(method,dfanno,genematrix,genesmore,allsamples){
    # This function structure was strongly suggested
    # by the Bioconductor reviewer
    supportedMethods<-c(
        "closest",
        "strongest",
        "sum",
        "topvar",
        "farthest",
        "lowvar"
    )
    if(!method%in%supportedMethods){
        stop("unsupported method ", method)
    }

    for (gene in genesmore) {
        subanno <- dfanno[dfanno$feature == gene, ]

        if (method == "closest") {
            closest <- which.min(subanno$distanceToStart)
            genematrix[gene, allsamples] <- as.numeric(subanno[closest,
                                                               allsamples])
        }

        if (method == "farthest") {
            farthest <- which.max(subanno$distanceToStart)
            genematrix[gene, allsamples] <- as.numeric(subanno[farthest,
                                                               allsamples])
        }

        if (method == "sum") {
            sums <- apply(subanno[, allsamples], 2, sum)
            genematrix[gene, allsamples] <- as.numeric(sums)
        }

        if (method == "strongest") {
            sums <- apply(subanno[, allsamples], 1, sum)
            top <- which.max(sums)
            genematrix[gene, allsamples] <- as.numeric(subanno[top,
                                                               allsamples])
        }

        if (method == "topvar") {
            vars <- apply(subanno[, allsamples], 1, var)
            top <- which.max(vars)
            genematrix[gene, allsamples] <- as.numeric(subanno[top,
                                                               allsamples])
        }

        if (method == "lowvar") {
            vars <- apply(subanno[, allsamples], 1, var)
            top <- which.min(vars)
            genematrix[gene, allsamples] <- as.numeric(subanno[top,
```

```
                                                                allsamples])
        }

    }
    return(genematrix)
}


vulcan.import.dba<- function (dbaobj, samples,intervals = NULL)
{
    dbcounts <- dbaobj
    listcounts <- dbcounts$peaks
    names(listcounts) <- dbcounts$samples[, 1]
    first <- listcounts[[1]]
    rawmat <- matrix(NA, nrow = nrow(first), ncol = length(listcounts) +
                         3)
    colnames(rawmat) <- c("Chr", "Start", "End", names(listcounts))
    rownames(rawmat) <- 1:nrow(rawmat)
    rawmat <- as.data.frame(rawmat)
    rawmat[, 1] <- as.character(first[, 1])
    rawmat[, 2] <- as.integer(first[, 2])
    rawmat[, 3] <- as.integer(first[, 3])
    for (i in 1:length(listcounts)) {
        rawmat[, names(listcounts)[i]] <- as.numeric(listcounts[[i]]$RPKM)
    }
    peakrpkms <- rawmat
    rm(rawmat)
    first <- listcounts[[1]]
    rawmat <- matrix(NA, nrow = nrow(first), ncol = length(listcounts) +
                         3)
    colnames(rawmat) <- c("Chr", "Start", "End", names(listcounts))
    rownames(rawmat) <- 1:nrow(rawmat)
    rawmat <- as.data.frame(rawmat)
    rawmat[, 1] <- as.character(first[, 1])
    rawmat[, 2] <- as.integer(first[, 2])
    rawmat[, 3] <- as.integer(first[, 3])
    for (i in 1:length(listcounts)) {
        rawmat[, names(listcounts)[i]] <- as.integer(listcounts[[i]]$Reads)
    }
    peakcounts <- rawmat
    rm(rawmat)
    vobj <- list(peakcounts = peakcounts, samples = samples,
                 peakrpkms = peakrpkms)
    return(vobj)
}

prependSampleNames <- function(vobj,prependString){
    colnames(vobj$peakcounts)[0:-3]<-paste0(prependString,colnames(vobj$peakcounts)[0:-3])
    vobj$samples[[2]]<-paste0(prependString,vobj$samples[[2]])
    vobj$samples[[1]]<-paste0(prependString,vobj$samples[[1]])
    colnames(vobj$peakrpkms)[0:-3]<-paste0(prependString,colnames(vobj$peakrpkms)[0:-3])
    return(vobj)
}
```

```
loadVulcanNetworks<-function(){
    regulons<-list()
    load("networks/laml-tf-regulon.rda")
    regulons$laml<-regul
    rm(regul)
    load("networks/brca-tf-regulon.rda")
    regulons$tcga<-regul
    rm(regul)
    load("networks/metabric-regulon-tfs.rda")
    regulons$metabric<-regulon
    rm(regulon)
    return(regulons)
}


#Slow so just load the file below
#vobj<-vulcan.import("samplesheet/samplesheet.csv")
#load(file="003_vobj.Rda")
samples <- list()
samples[['ER']]<-c('1a','2a','3a')
samples[['none']]<-c('1b','2b','3b')

vobj <-vulcan.import.dba(GRHL2,samples)

#vobj<-vulcan.annotate(vobj,lborder=-10000,rborder=10000,method='sum')

vobj <-prependSampleNames(vobj,"X")

lborder=-10000
rborder=10000
method='sum'
#DEBUG
#source("https://bioconductor.org/biocLite.R")
#biocLite("TxDb.Hsapiens.UCSC.hg38.knownGene")
suppressMessages(library("TxDb.Hsapiens.UCSC.hg38.knownGene"))

annotation <- toGRanges(TxDb.Hsapiens.UCSC.hg38.knownGene,
                        feature = "gene")
gr <- GRanges(vobj$peakcounts)
seqlevels(annotation)<- sub('chr','',seqlevels(annotation))
anno <- annotatePeakInBatch(gr, AnnotationData = annotation,
                            output = "overlapping", FeatureLocForDistance = "TSS",
                            bindingRegion = c(lborder, rborder))
```

```
## Annotate peaks by annoPeaks, see ?annoPeaks for details.
```

```
## maxgap will be ignored.
```

```
dfanno <- anno
names(dfanno) <- seq_len(length(dfanno))
dfanno <- as.data.frame(dfanno)
allsamples <- unique(unlist(vobj$samples))
genes <- unique(dfanno$feature)
peakspergene <- table(dfanno$feature)
```

```
rawcounts <- matrix(NA, nrow = length(genes), ncol = length(allsamples))
colnames(rawcounts) <- allsamples
rownames(rawcounts) <- genes
genesone <- names(peakspergene)[peakspergene == 1]
for (gene in genesone) {
    rawcounts[gene, allsamples] <- as.numeric(dfanno[dfanno$feature ==
                                                         gene, allsamples])
}

genesmore <- names(peakspergene)[peakspergene > 1]
rawcounts <- dist_calc(method, dfanno, rawcounts, genesmore,
                       allsamples)

gr <- GRanges(vobj$peakrpkms)
anno <- annotatePeakInBatch(gr, AnnotationData = annotation,
                            output = "overlapping", FeatureLocForDistance = "TSS",
                            bindingRegion = c(lborder, rborder))
```

```
## Annotate peaks by annoPeaks, see ?annoPeaks for details.
## maxgap will be ignored.
```

```
dfanno <- anno
names(dfanno) <- seq_len(length(dfanno))
dfanno <- as.data.frame(dfanno)
allsamples <- unique(unlist(vobj$samples))
genes <- unique(dfanno$feature)
peakspergene <- table(dfanno$feature)
rpkms <- matrix(NA, nrow = length(genes), ncol = length(allsamples))
rownames(rpkms) <- genes
genesone <- names(peakspergene)[peakspergene == 1]
colnames(rpkms)<-allsamples
for (gene in genesone) {
    rpkms[gene, allsamples] <- as.numeric(dfanno[dfanno$feature ==
                                                     gene, allsamples])
}
genesmore <- names(peakspergene)[peakspergene > 1]

rpkms <- dist_calc(method, dfanno, rpkms, genesmore, allsamples)
rawcounts <- matrix(as.numeric(rawcounts), nrow = nrow(rawcounts),
                    dimnames = dimnames(rawcounts))
rpkms <- matrix(as.numeric(rpkms), nrow = nrow(rpkms), dimnames = dimnames(rpkms))
vobj$rawcounts <- rawcounts
colnames(rpkms)<-allsamples
vobj$rpkms <- rpkms

#DEBUG ENDS

vobj<-vulcan.normalize(vobj)

load(file="networks/metabric-regulon-tfs.rda")

regulons <- loadVulcanNetworks()
suppressMessages(library("org.Hs.eg.db"))
list_eg2symbol<-as.list(org.Hs.egSYMBOL[mappedkeys(org.Hs.egSYMBOL)])
```

```
vobj_results<-list()

#test<-vulcan(vobj,       network=regulon, contrast=c("none","ER")    )

networks<-c("tcga","metabric") #,"laml")

for (network in networks) {
    vobj_results[[network]]<-vulcan(vobj,
                                    network=regulons[[network]],
                                    contrast=c("none","ER"),
                                    annotation=list_eg2symbol)
}
```

```
## Thu Jan 25 17:38:45 2018
## Computing the null model distribution by 1000 permutations.
## ------------------------------------------------------------
## Thu Jan 25 17:39:01 2018
## Computing regulon enrichment with aREA algorithm
##
## Estimating the normalized enrichment scores
## Thu Jan 25 17:39:38 2018
## Computing the null model distribution by 1000 permutations.
## ------------------------------------------------------------
## Thu Jan 25 17:39:53 2018
## Computing regulon enrichment with aREA algorithm
##
## Estimating the normalized enrichment scores
```

```
vobj_objects<-list(vobj_results)
names(vobj_objects)<-c("GRHL2")
networks<-c("tcga","metabric")
```

Transcription factor activity

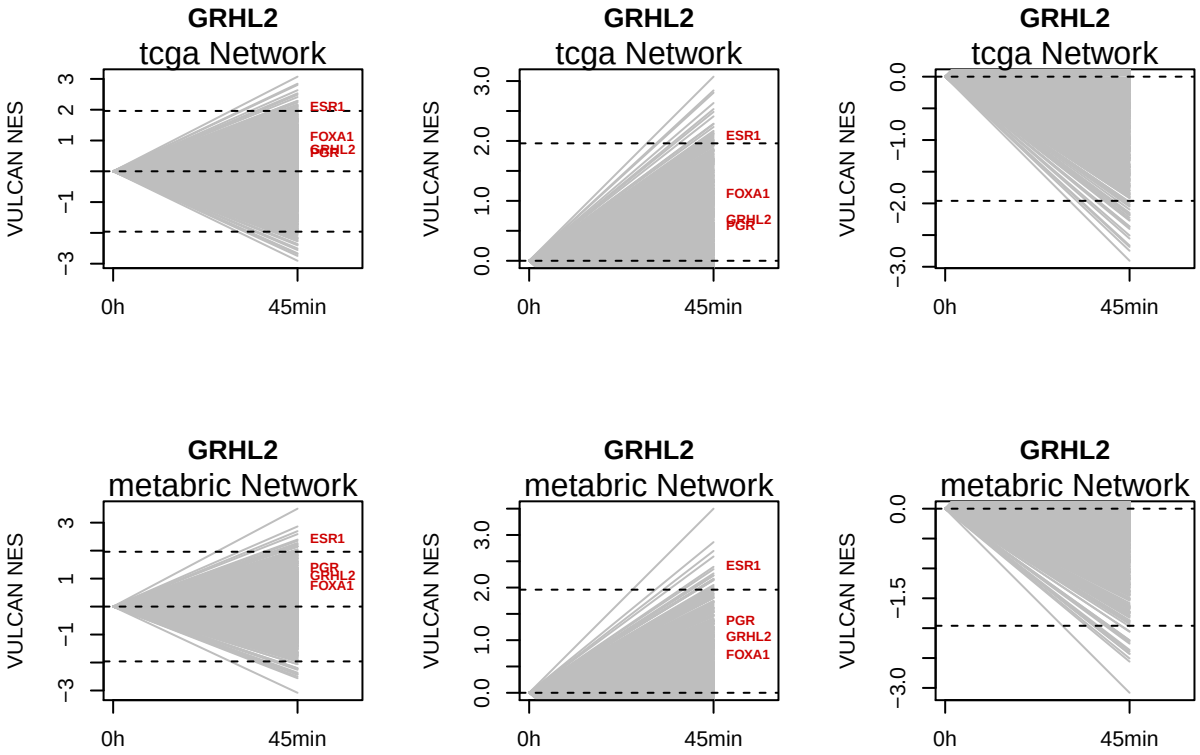
```
plotVulcan <-function(vobj,threshold,network_title,title,plotTF,xlim,ylim) {
    threshold<-sign(threshold)*p2z(abs(threshold))
    network=vobj$mrs[,"NES"]
    tfs<-names(network)
    networkmat<-cbind(rep(0,length(network)),network[tfs])
    colnames(networkmat)<-c("0h","45min")
    matplot(t(networkmat),type="l",col="grey",ylab="VULCAN NES",xaxt="n",lty=1,main=title,xlim=xlim,yli
    axis(1,at=c(1:2),labels=colnames(networkmat))
    abline(h=c(0,threshold,-threshold),lty=2)
    text(2,networkmat[plotTF,2],label=names(networkmat[,2][plotTF]),pos=4,cex=0.6,font=2,col="red3")
    mtext(network_title)
}

par(mfrow=c(2,3))
for(network in networks){
    for (vobj_name in names(vobj_objects))
    {
```

```
        vobj<-vobj_objects[[vobj_name]]
        vobj<-vobj[[network]]
        #TFs<-getTFs(vobj)
        TFs=c("ESR1","PGR","FOXA1","GRHL2") #Overide TFS
        plotVulcan(vobj,0.05, paste0(network," Network"),vobj_name,TFs,xlim=c(1,2.3), ylim=c(min(vobj$m

        #TFs<-getTFs(vobj,0.05)
        plotVulcan(vobj,0.05, paste0(network," Network"),vobj_name,TFs,xlim=c(1,2.3), ylim=c(0,max(vobj

        #TFs<-getTFs(vobj,-0.05)
        plotVulcan(vobj,-0.05, paste0(network," Network"),vobj_name,TFs,xlim=c(1,2.3), ylim=c(min(vobj$m
    }
}
```



METABRIC TGCA comparison

```
interect<-intersect(rownames(vobj_results$metabric$mrs),
                    rownames(vobj_results$tcga$mrs))
plot(-log10(vobj_results$metabric$mrs[interect,"pvalue"]),
     -log10(vobj_results$tcga$mrs[interect,"pvalue"]),
     pch=20, xlab="Metabric Network Enrichment Score", ylab="TCGA Network Enrichment Score",main="VULCA

filtered<-(log10(vobj_results$metabric$mrs[interect,"pvalue"])^2+log10(vobj_results$tcga$mrs[interect,"

text(-log10(vobj_results$metabric$mrs[interect,"pvalue"][filtered]),
     -log10(vobj_results$tcga$mrs[interect,"pvalue"][filtered])-0.1,
     labels=interect[filtered])
```

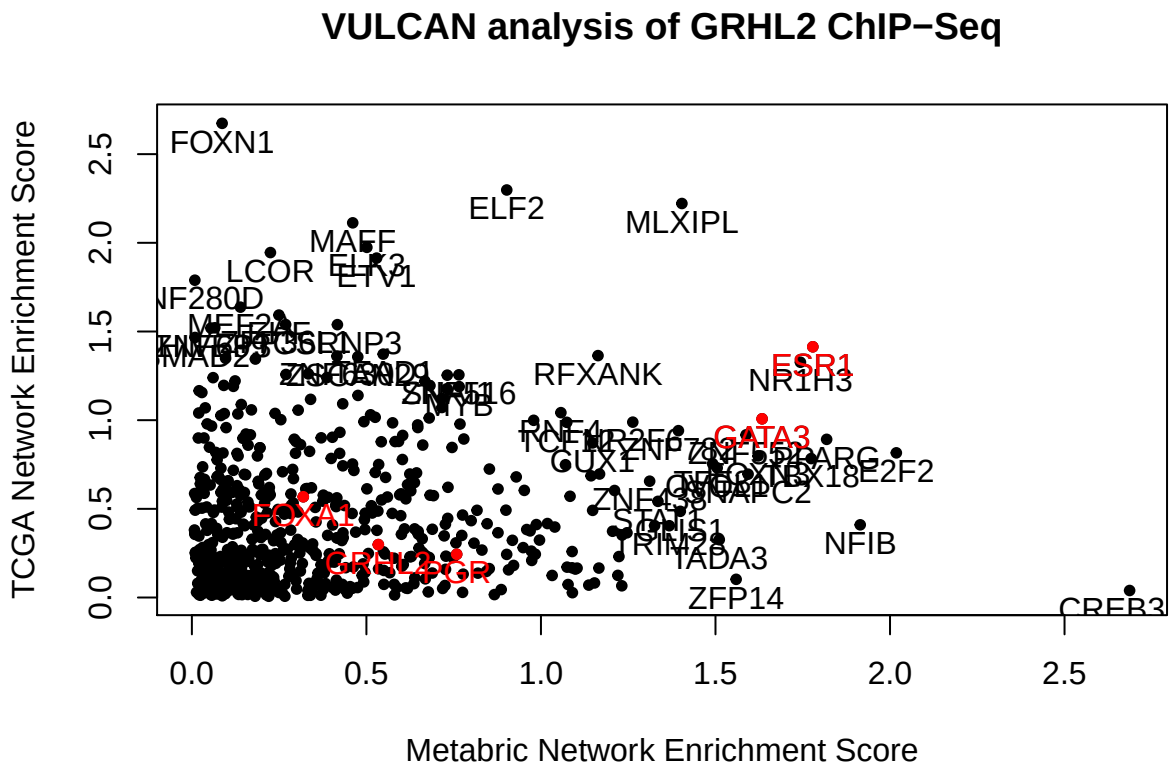


Figure 5: Comparision of results of VULCAN analysis for METABRIC and TGCA

```
points(-log10(vobj_results$metabric$mrs[interect,"pvalue"][c("GATA3","PGR","FOXA1","ESR1","GRHL2")]),
       -log10(vobj_results$tcga$mrs[interect,"pvalue"][c("GATA3","PGR","FOXA1","ESR1","GRHL2")]),pch=20
text(-log10(vobj_results$metabric$mrs[interect,"pvalue"][c("GATA3","PGR","FOXA1","ESR1","GRHL2")]),
     -log10(vobj_results$tcga$mrs[interect,"pvalue"][c("GATA3","PGR","FOXA1","ESR1","GRHL2")])-0.1,col=
     labels=c("GATA3","PGR","FOXA1","ESR1","GRHL2"))
```

Motif Analysis

Export bed files

```
#Note homer needs this as Hg format so you need to change from "1" to "chr1" etc.
df<-as.data.frame(dba.report(GRHL2))
df$seqnames<-paste0("chr",df$seqnames)

kable(head(df))
```

	seqnames	start	end	width	strand	Conc	Conc_none	Conc_ER	Fold	p.value	FDR
12123	chr14	93959394	93959894	501	*	7.63	6.04	8.37	-2.33	0	0e+00
9047	chr12	75706270	75706770	501	*	6.74	4.93	7.52	-2.60	0	0e+00
24493	chr21	31529399	31529899	501	*	6.50	4.86	7.25	-2.39	0	0e+00
129	chr1	7447769	7448269	501	*	4.98	2.44	5.85	-3.41	0	0e+00
40054	chr9	75101020	75101520	501	*	6.94	4.80	7.76	-2.96	0	0e+00
28564	chr3	176953741	176954241	501	*	5.53	3.80	6.30	-2.50	0	1e-07

```
#write.bed(df,"bed/up.bed")
```



```
df_all<-as.data.frame(dba.report(GRHL2, th=1))
df_all$seqnames<-paste0("chr",df_all$seqnames)

kable(head(df_all))
```

```
cd motifAnalysis
findMotifsGenome.pl ../bed/up.bed hg38 grhl2UpSites
findMotifsGenome.pl ../bed/all.bed hg38 grhl2AllSites
```

Generate promoter locations

Generate Bed file of Promoters

```
require(TxDb.Hsapiens.UCSC.hg38.knownGene)
txdb <- TxDb.Hsapiens.UCSC.hg38.knownGene
p<-promoters(genes(txdb), upstream = 1500, downstream = 500)

write.bed<-function(df, filename){
    write.table(as.data.frame(df)[,1:3],
                filename, quote=FALSE, sep="\t",
                row.names=FALSE, col.names=FALSE)
}

write.bed(p,"bed/promoter.bed")
```

Ovlerap of GRHL2 binding sites with published data

Extract ChIA-PET Data

```
#ChIA-pet all 3 https://www.encodeproject.org/experiments/ENCSR000BZZ/
cd bed
multiIntersectBed -i *bed6_sorted.bed | awk '{if ($4 > 1) {print} }' > hglft_ChIA_combined.bed
```

Find overlapping sites

```
#p300 from Zwart, EMBO, 2011
#ER from Hurtado 2011
#gro-seq GSE43835
#Rest from Caroll MLL3 paper

cd bed
mkdir overlaps

#generate gro-seq interect
#file1=hglft_E2.40m.rep1.bed
#file2=hglft_E2.40m.rep2.bed
#output=gro-seq.bed
#bedtools intersect -sorted -a $file1 -b $file2 > overlaps/$output

file1=hglft_foxa1.bed
file2=up.bed
output=foxa1-up.bed
bedtools intersect -a $file1 -b $file2 > overlaps/$output

file1=hglft_ER_Hurtado_2011.bed
```

```
output=er-up.bed
bedtools intersect -a $file1 -b $file2 > overlaps/$output

#file1=gro-seq.bed
#output=gro-up.bed
#bedtools intersect -a $file1 -b $file2 > overlaps/$output


file1=hglft_ChIA_combined.bed
output=ChIAPet_up.bed
bedtools intersect -a $file1 -b $file2 > overlaps/$output

file1=hglft_h3k4me1.bed
output=h3k4me1-up.bed
bedtools intersect -a $file1 -b $file2 > overlaps/$output

file1=hglft_h3k4me3.bed
output=h3k4me3-up.bed
bedtools intersect -a $file1 -b $file2 > overlaps/$output

file1=hglft_p300_ctrl.bed
output=p300_ctrl_up.bed
bedtools intersect -a $file1 -b $file2 > overlaps/$output

file1=hglft_p300_e2.bed
output=p300_e2_up.bed
bedtools intersect -a $file1 -b $file2 > overlaps/$output


#all peaks

file1=hglft_foxa1.bed
file2=all.bed
output=foxa1-all.bed
bedtools intersect -a $file1 -b $file2 > overlaps/$output

file1=hglft_ER_Hurtado_2011.bed
output=er-all.bed
bedtools intersect -a $file1 -b $file2 > overlaps/$output

#file1=gro-seq.bed
#output=gro-all.bed
#bedtools intersect -a $file1 -b $file2 > overlaps/$output

file1=hglft_ChIA_combined.bed
output=ChIAPet_all.bed
bedtools intersect -a $file1 -b $file2 > overlaps/$output

file1=hglft_h3k4me1.bed
output=h3k4me1-all.bed
bedtools intersect -a $file1 -b $file2 > overlaps/$output

file1=hglft_h3k4me3.bed
```

```
output=h3k4me3-all.bed
bedtools intersect -a $file1 -b $file2 > overlaps/$output

file1=hglft_p300_ctrl.bed
output=p300_ctrl_all.bed
bedtools intersect -a $file1 -b $file2 > overlaps/$output

file1=hglft_p300_e2.bed
output=p300_e2_all.bed
bedtools intersect -a $file1 -b $file2 > overlaps/$output

##Overlap with promoters

file1=overlaps/ChIAPet_up.bed
file2=promoter.bed
output=ChIAPet_up_prom.bed
bedtools intersect -a $file1 -b $file2 > overlaps/$output

file1=overlaps/ChIAPet_all.bed
file2=promoter.bed
output=ChIAPet_all_prom.bed
bedtools intersect -a $file1 -b $file2 > overlaps/$output
```

Count number of overlaping sites

```
cd bed/overlaps
wc -l *.bed > overlaps.txt
```

Table of number of overlaping sites for each factor

```
overlaps<-read.table("bed/overlaps/overlaps.txt")
kable(overlaps)
```

V1	V2
1764	ChIAPet_all.bed
109	ChIAPet_all_prom.bed
214	ChIAPet_up.bed
8	ChIAPet_up_prom.bed
5539	er-all.bed
318	er-up.bed
13422	foxa1-all.bed
255	foxa1-up.bed
33831	h3k4me1-all.bed
343	h3k4me1-up.bed
10234	h3k4me3-all.bed
55	h3k4me3-up.bed
7072	p300_ctrl_all.bed
155	p300_ctrl_up.bed
13588	p300_e2_all.bed
337	p300_e2_up.bed

V1	V2
87244	total

GRHL2 binding overlap with known factors

```
overlapsDF<-cbind(overlaps[1:16,],c('all','all','up','up','all','up','all','up','all','up','all','up','
overlapsDF<-cbind(overlapsDF,c('all','promoter','all','promoter',rep("all",12)))
overlapsDF<-cbind(overlapsDF,c(rep("None",14),"E2","E2"))
colnames(overlapsDF)<-c("Number","Factor","GRHL2","Feature","Treatment")
overlapsDF$Factor<-c(rep("ChIAPet",4),rep("ER",2),rep("FOXA1",2),rep("H3K4Me1",2),rep("H3K4Me3",2),rep(
overlapsDF$Number<-as.numeric(overlapsDF$Number)
ChIAPet<-overlapsDF[overlapsDF$Factor=='ChIAPet' & overlapsDF$Feature=='all',]
ChIAPet$Number<-overlapsDF[overlapsDF$Factor=='ChIAPet' & overlapsDF$Feature=='all',]$Number - overlapsD
ChIAPet$Feature<-c('enchancer','enchancer')
overlapsDF<-rbind(overlapsDF,ChIAPet)
overlapsDF<-data.frame(overlapsDF)
overlapsDF[overlapsDF$GRHL2=='up',]$Number<-(overlapsDF[overlapsDF$GRHL2=='up',]$Number/355)*100 #355 i
overlapsDF[overlapsDF$GRHL2=='all',]$Number<-(overlapsDF[overlapsDF$GRHL2=='all',]$Number/42721)*100 #T

library(lattice)

barchart(Number ~ Factor ,data=overlapsDF[overlapsDF$Factor != 'ChIAPet',],
         par.settings = simpleTheme(col=c("lightblue","pink")),
         group=GRHL2,auto.key=list(space="top", columns=2, text=c('All GRHL2 Sites','GRHL2 Sites Respons
         main="GRHL2 Binding Overlap with Known Factors", ylab="Percentage of GRHL2 sites that overlap"
         xlab="Factor")
```

GRHL2 binding overlap with P300 binding sites

```
barchart(Number ~ Treatment ,data=overlapsDF[overlapsDF$Factor=='P300',],
         group=GRHL2,
         auto.key=list(space="top",columns=2, text=c('All GRHL2 Sites','GRHL2 Responsive to E2')),
         par.settings = simpleTheme(col=c("lightblue","pink")),
         main="GRHL2 overlap with P300 binding sites",
         xlab="P300 ChIP-Seq Condition",
         ylab="Percentage of GRHL2 sites that overlap")
```

GRHL2 overlap with ChIA-PET sites

```
barchart(Number ~ Feature ,data=overlapsDF[overlapsDF$Factor=='ChIAPet',],
         group=GRHL2,
         auto.key=list(space="top",columns=2, text=c('All GRHL2 Sites','GRHL2 Sites Responsive to E2'))
         par.settings = simpleTheme(col=c("lightblue","pink")),
         main="GRHL2 overlap with ChIA-PET sites",
         xlab="Genomic Feature ChIP-Seq Condition",
         ylab="Percentage of GRHL2 sites that overlap")
```

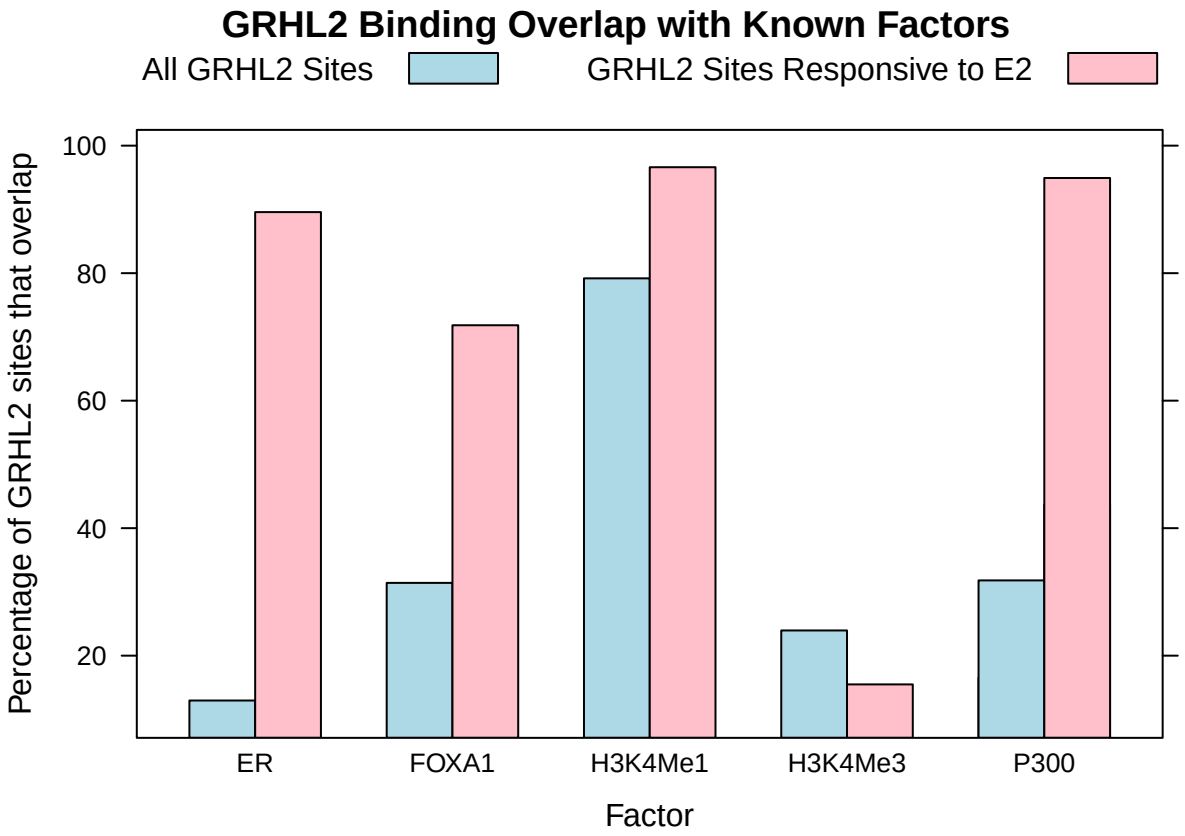


Figure 6: GRHL2 binding overlap with known factors

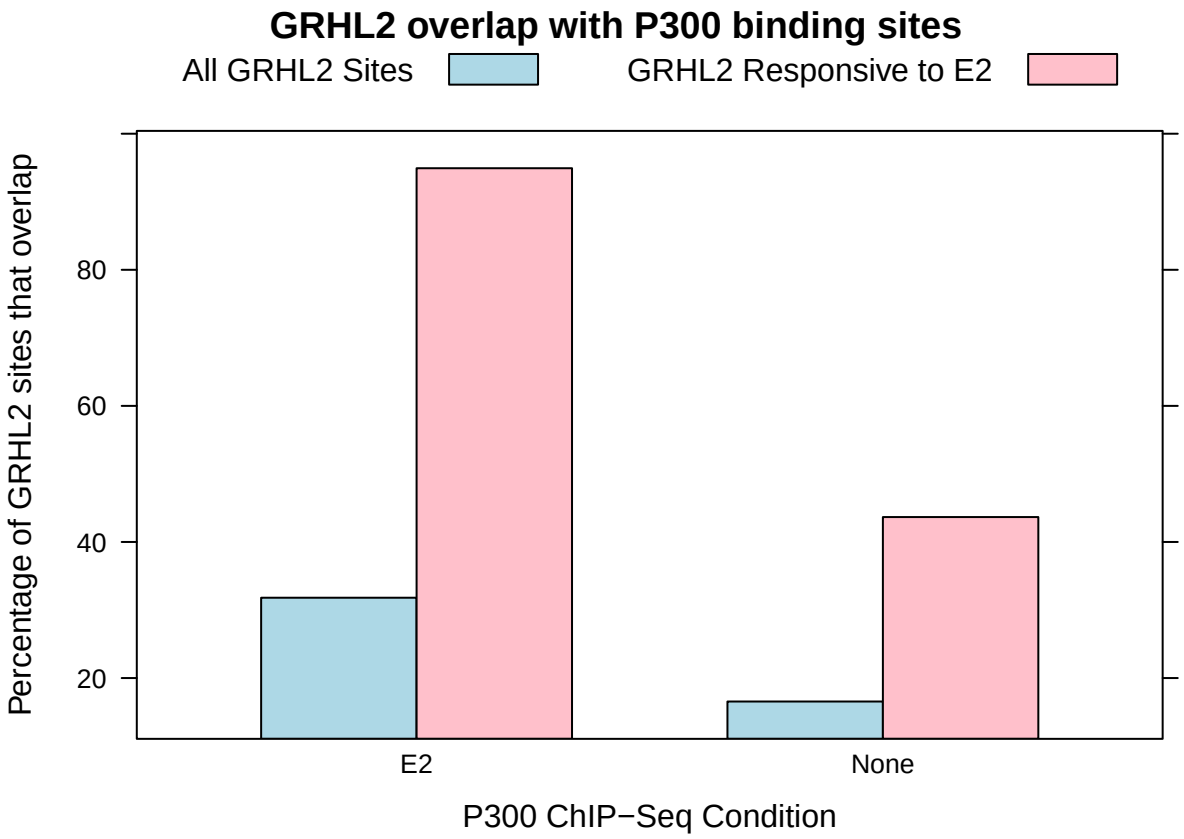


Figure 7: GRHL2 binding overlap with P300 binding sites

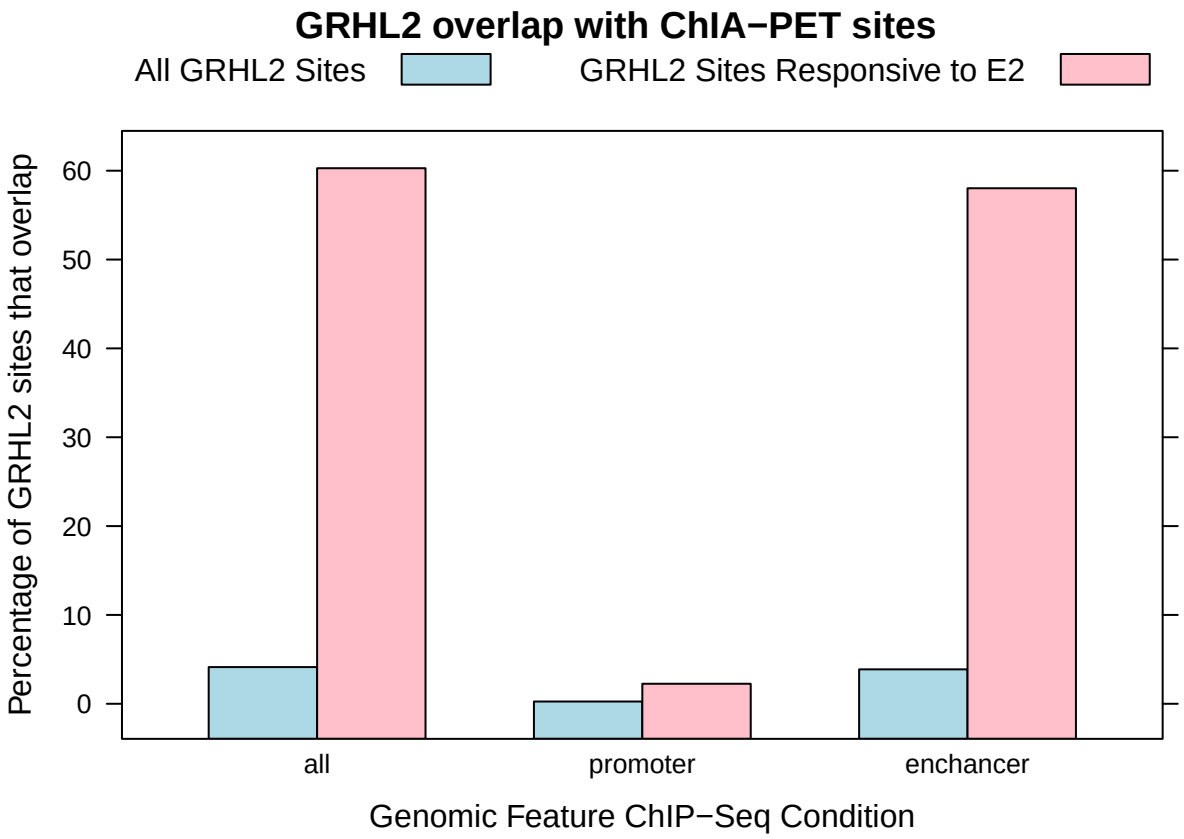


Figure 8: GRHL2 overlap with ChIA-PET sites

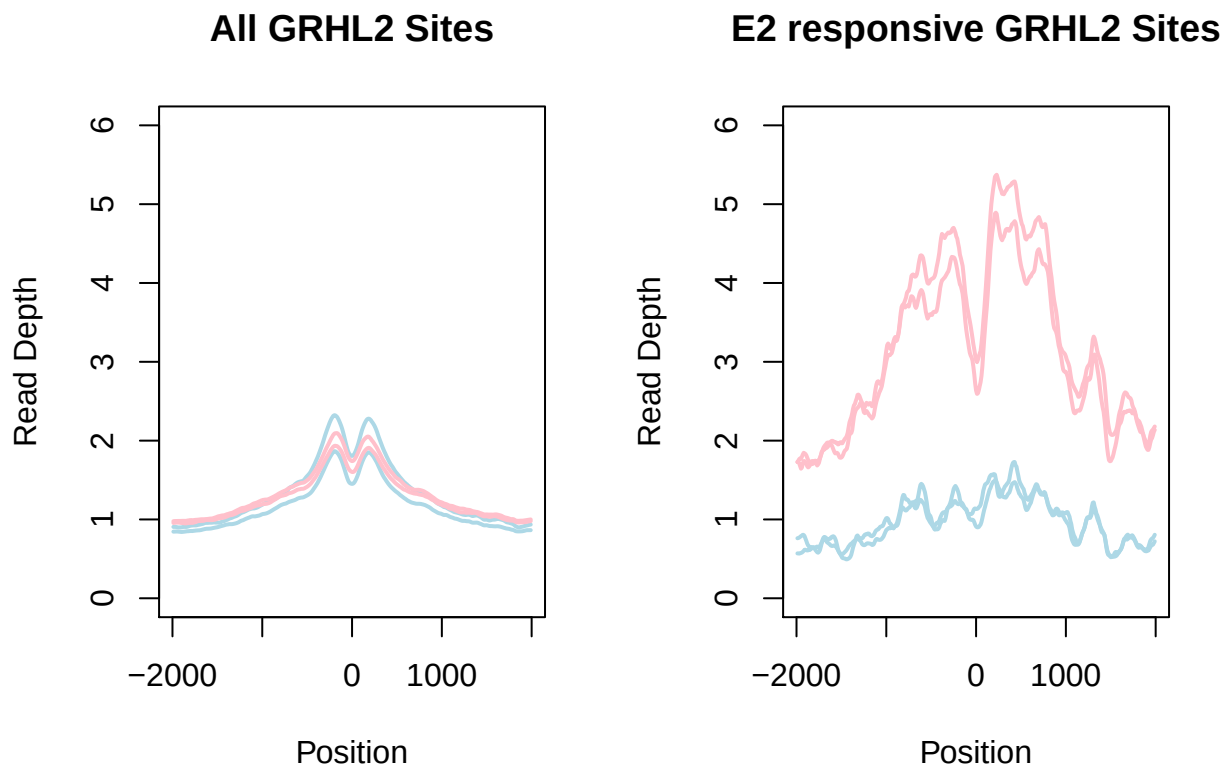


Figure 9: Gro-Seq data at GRHL2 sites. Blue is control samples, pink is E2 treated samples. Data from GSE43836.

GRHL2 overlap with Gro-SEQ data

```
allPeaks<-read.csv("txt/All_GRHL2_peaks_GSE43836.csv")
upPeaks<-read.csv("txt/Up_GRHL2_peaks_GSE43836.csv")


par(mfrow=c(1,2))

plot(allPeaks[1:2],ylim=c(0,6),type="n", lwd=2, ylab="Read Depth", main="All GRHL2 Sites")
lines(allPeaks[c(1,2)], lwd=2, col="lightblue")
lines(allPeaks[c(1,3)], lwd=2,  col="lightblue")
lines(allPeaks[c(1,4)], lwd=2,  col="pink")
lines(allPeaks[c(1,5)],  lwd=2, col="pink")

plot(upPeaks[c(1,5)],ylim=c(0,6),type="n", ylab="Read Depth", main="E2 responsive GRHL2 Sites")
lines(upPeaks[c(1,2)], lwd=2,col="lightblue")
lines(upPeaks[c(1,3)], lwd=2, col="lightblue")
lines(upPeaks[c(1,4)], lwd=2, col="pink")
lines(upPeaks[c(1,5)],  lwd=2,col="pink")
```