Analysis of ZNF143 depletion HEK-293T PRO-seq data

Sathyan Kizhakke Mattada*, Warren Anderson†, Michael J. Guertin‡

February 20, 2019

Abstract

Rapid perturbation of protein function permits the ability to define primary molecular responses without the need to consider downstream cumulative effects of protein dysregulation. The auxin-inducible degron system was developed as a tool to achieve rapid and inducible protein degradation in non-plant systems. However, we found that tagging transcription factors at their endogenous loci results in chronic auxin-independent degradation by the proteasome. We rescued the chronic degradation phenotype by co-expressing the Phox and Bem1 (PB1) domain of an *Oryza sativa* Auxin Response Transcription Factor (ARF). ARF interacts with the auxin-inducible degron (AID) to mediate protein stability. Degradation kinetics is substantially faster upon ARF rescue. We also observed that auxin, an aromatic plant hormone, activates aryl hydrocarbon receptor genes in human cell lines. We endogenously tagged the ZNF143 transcription factor with AID and rescued chronic depletion with ARF expression. We performed nascent RNA sequencing (PRO-seq) after ZNF143 depletion to identify primary ZNF143 response genes and to define the molecular function of ZNF143. We found that ZNF143 exclusively activates genes in *cis* and controls promoter-proximal paused RNA Polymease II (PolII) density. Mathematical modeling of changes in PolII density in the pause region and the gene body indicates that ZNF143 functions in PolII recruitment/initiation or to stabilize promoter-proximal paused PolII.

Contents

*sk8fz@virginia.edu
†wa3j@virginia.edu section 7
‡guertin@virginia.edu

List of Figures

1 Processing PRO-seq reads

1.1 Processing FASTQ files/Aligning

All the files should be named with the cell type, genetic modifications, description, the replicate number, and then the paired end number. No spaces. Subsequent `shell` scripts rely on this nomenclature.

We performed paired-end sequencing and we can use the 5 prime end of the nascent RNAs to call transcription start sites *de novo*. However, these analyses are not included adn the PE2 reads are ignored.

```
cd ~
mkdir ZNF143
cd ZNF143

#retrieve and split SRA file into PE fastq files
#update this section once the SRAfile names are assigned
```

We first use `cutadapt` to discard reads fewer than 18 bases (Martin, 2011). Note that we have an 8 base unique molecular identifier (UMI) to filter out PCR duplicates–therefore, the `-m` command value is 26. Recall that the 5 prime adapter sequence for PRO-seq is: CCUUGGCACCCGAGAAUUCCA (Mahat *et al.*, 2016) `https://www.nature.com/articles/nprot.2016.086/tables/1`

Next we use `fqdedup` from `https://github.com/guertinlab/fqdedup` to remove PCR duplicates.

We then implement `fastx_tools` to trim off the UMI, constrain the longest read length to be 30 bases, and reverse complement the read `https://github.com/agordon/fastx_toolkit`.

We then align to the `hg38` genome using bowtie2 (Langmead *et al.*, 2009) and convert the sam file to the compressed and sorted BAM format using `samtools` (Li *et al.*, 2009).

We are only aligning the paired end 1 read, but we performed paired end sequencing to have the ability to go back to the PE2 read and infer transcription start sites from 5 prime end read pile ups.

1.2 Separate by strand, shift, and convert to `bigWig`

First we separate the `BAM` by strand. Here we are not implementing enzyme bias scaling of the PRO-seq reads, but `seqOutBias` (Martins *et al.*, 2018) has some useful features (e.g. –shift-counts) and mappability filtering, so we will use `seqOutBias` to make `bigWig` files (Martins *et al.*, 2018).

```
fastq-dump --version
cutadapt --version
fqdedup --version
fastx_trimmer -h
fastx_reverse_complement -h
bowtie2 --version
samtools --version
seqOutBias --version
python --version
bedGraphToBigWig
macs2 --version
bedtools --version
meme -version
tomtom -version
bigWigMerge

#input files
ls *PE1.fastq.gz
ls HEK2*.fastq.gz

##
## fastq-dump : 2.8.2
##
## 1.14
## fqdedup v0.1.0
## usage: fastx_trimmer [-h] [-f N] [-l N] [-t N] [-m MINLEN] [-z] [-v] [-i INFILE] [-o OUTFILE]
## Part of FASTX Toolkit 0.0.14 by A. Gordon (assafgordon@gmail.com)
##
##    [-h]         = This helpful help screen.
##    [-f N]       = First base to keep. Default is 1 (=first base).
##    [-l N]       = Last base to keep. Default is entire read.
##    [-t N]       = Trim N nucleotides from the end of the read.
##                   '-t'  can not be used with '-l' and '-f'.
##    [-m MINLEN]  = With [-t], discard reads shorter than MINLEN.
##    [-z]         = Compress output with GZIP.
##    [-i INFILE]  = FASTA/Q input file. default is STDIN.
##    [-o OUTFILE] = FASTA/Q output file. default is STDOUT.
##
## usage: fastx_reverse_complement [-h] [-r] [-z] [-v] [-i INFILE] [-o OUTFILE]
## Part of FASTX Toolkit 0.0.14 by A. Gordon (assafgordon@gmail.com)
##
##    [-h]         = This helpful help screen.
##    [-z]         = Compress output with GZIP.
##    [-i INFILE]  = FASTA/Q input file. default is STDIN.
##    [-o OUTFILE] = FASTA/Q output file. default is STDOUT.
##
## /usr/local/bin/../Cellar/bowtie2/2.3.2/bin/bowtie2-align-s version 2.3.2
## 64-bit
## Built on Sierra.local
## Tue May  9 08:11:47 BST 2017
## Compiler: InstalledDir: /Applications/Xcode.app/Contents/Developer/Toolchains/XcodeDefault.xctoolchain/usr/bin
## Options: -O3 -m64 -msse2 -funroll-loops -g3 -L /usr/local/lib -DPOPCNT_CAPABILITY -DWITH_TBB -DNO_SPINLOCK -DWITH_QUEUELOCK=1
## Sizeof {int, long, long long, void*, size_t, off_t}: {4, 8, 8, 8, 8, 8}
## samtools 1.6
## Using htslib 1.6
## Copyright (C) 2017 Genome Research Ltd.
## Cut-site frequencies, v1.1.3
## Python 2.7.10
## bedGraphToBigWig v 4 - Convert a bedGraph file to bigWig format.
```

```
## usage:
##    bedGraphToBigWig in.bedGraph chrom.sizes out.bw
## where in.bedGraph is a four column file in the format:
##       <chrom> <start> <end> <value>
## and chrom.sizes is a two-column file/URL: <chromosome name> <size in bases>
## and out.bw is the output indexed big wig file.
## If the assembly <db> is hosted by UCSC, chrom.sizes can be a URL like
##   http://hgdownload.cse.ucsc.edu/goldenPath/<db>/bigZips/<db>.chrom.sizes
## or you may use the script fetchChromSizes to download the chrom.sizes file.
## If not hosted by UCSC, a chrom.sizes file can be generated by running
## twoBitInfo on the assembly .2bit file.
## The input bedGraph file must be sorted, use the unix sort command:
##   sort -k1,1 -k2,2n unsorted.bedGraph > sorted.bedGraph
## options:
##    -blockSize=N - Number of items to bundle in r-tree.  Default 256
##    -itemsPerSlot=N - Number of data points bundled at lowest level. Default 1024
##    -unc - If set, do not use compression.
## macs2 2.1.1.20160309
## bedtools v2.27.1
## 5.0.1
## 5.0.1
## bigWigMerge v2 - Merge together multiple bigWigs into a single output bedGraph.
## You'll have to run bedGraphToBigWig to make the output bigWig.
## The signal values are just added together to merge them
## usage:
##    bigWigMerge in1.bw in2.bw .. inN.bw out.bedGraph
## options:
##    -threshold=0.N - don't output values at or below this threshold. Default is 0.0
##    -adjust=0.N - add adjustment to each value
##    -clip=NNN.N - values higher than this are clipped to this value
##    -inList - input file are lists of file names of bigWigs
##    -max - merged value is maximum from input files rather than sum
##
## HEK_TIR1_ZNF143AID_ARF_Auxin_rep1_PE1.fastq.gz
## HEK_TIR1_ZNF143AID_ARF_Auxin_rep2_PE1.fastq.gz
## HEK_TIR1_ZNF143AID_ARF_Auxin_rep3_PE1.fastq.gz
## HEK_TIR1_ZNF143AID_ARF_Auxin_rep4_PE1.fastq.gz
## HEK_TIR1_ZNF143AID_ARF_rep1_PE1.fastq.gz
## HEK_TIR1_ZNF143AID_ARF_rep2_PE1.fastq.gz
## HEK_TIR1_ZNF143AID_ARF_rep3_PE1.fastq.gz
## HEK_TIR1_ZNF143AID_ARF_rep4_PE1.fastq.gz
## HEK_TIR1_ZNF143AID_Auxin_rep1_PE1.fastq.gz
## HEK_TIR1_ZNF143AID_Auxin_rep2_PE1.fastq.gz
## HEK_TIR1_ZNF143AID_Auxin_rep3_PE1.fastq.gz
## HEK_TIR1_ZNF143AID_Auxin_rep4_PE1.fastq.gz
## HEK_TIR1_ZNF143AID_rep1_PE1.fastq.gz
## HEK_TIR1_ZNF143AID_rep2_PE1.fastq.gz
## HEK_TIR1_ZNF143AID_rep3_PE1.fastq.gz
## HEK_TIR1_ZNF143AID_rep4_PE1.fastq.gz
## HEK_TIR1_rep1_PE1.fastq.gz
## HEK_TIR1_rep2_PE1.fastq.gz
## HEK_TIR1_rep3_PE1.fastq.gz
## HEK_TIR1_rep4_PE1.fastq.gz
## HEK293T_TIR1_Cl4_3hrAUXIN_rep1.fastq.gz
## HEK293T_TIR1_Cl4_3hrAUXIN_rep2.fastq.gz
## HEK293T_TIR1_Cl4_3hrDMSO_rep1.fastq.gz
## HEK293T_TIR1_Cl4_3hrDMSO_rep2.fastq.gz
```

```
wget https://hgdownload-test.gi.ucsc.edu/goldenPath/hg38/bigZips/hg38.chrom.sizes
wget http://hgdownload.cse.ucsc.edu/goldenPath/hg38/bigZips/hg38.fa.gz
gunzip hg38.fa.gz
bowtie2-build hg38.fa hg38



for i in *PE1.fastq.gz
do
    name=$(echo $i | awk -F"/" '{print $NF}' | \
           awk -F".fastq.gz" '{print $1}')
    echo $name
    cutadapt -m 26 -a TGGAATTCTCGGGTGCCAAGG ${name}.fastq.gz | \
       fqdedup -i - -o - | \
       fastx_trimmer -f 9 -l 38 | \
       fastx_reverse_complement -z -o ${name}.processed.fastq.gz
    bowtie2 -p 3 -x hg38 -U ${name}.processed.fastq.gz | \
       samtools view -b - | \
       samtools sort - -o ${name}.sorted.bam
       samtools view -bh -F 20 ${name}.sorted.bam > ${name}_pro_plus.bam
```

```
       samtools view -bh -f 0x10 ${name}.sorted.bam > ${name}_pro_minus.bam
       seqOutBias hg38.fa ${name}_pro_plus.bam --no-scale --skip-bed \
          --bw=${name}_plus_body_0-mer.bigWig --tail-edge --read-size=30
       seqOutBias hg38.fa ${name}_pro_minus.bam --no-scale --skip-bed \
           --bw=${name}_minus_body_0-mer.bigWig --tail-edge --read-size=30
done
```

1.3 Calling TSS coordinates

Assigning predominant transcription start sites is an important step when analyzing PRO-seq data. We will start with all Ensembl annotations and choose the one with highest promoter proximal pausing density using all HEK-293T merged data. The following code retrieves the gene annotation (`GTF`) file, extracts all annotated transcription start sites (TSS), combines all data into a single `bigWig` per strand, and extracts all genic annotations.

```
#retrieve gene annotation file
wget ftp://ftp.ensembl.org/pub/release-87/gtf/homo_sapiens/Homo_sapiens.GRCh38.87.gtf.gz
gunzip Homo_sapiens.GRCh38.87.gtf.gz

#parse all TSS--exons 1
grep 'exon_number "1"' Homo_sapiens.GRCh38.87.gtf | \
    sed 's/^/chr/' | \
    awk '{OFS="\t";} {print $1,$4,$5,$14,$20,$7}' | \
    sed 's/";//g' | \
    sed 's/"//g' > Homo_sapiens.GRCh38.87.tss.bed

#combine all PRO data from HEK

pfiles=$(ls *PE1*plus*bam)
mfiles=$(ls *PE1*minus*bam)

seqOutBias hg38.fa $pfiles --no-scale --bw=HEK_plus_combined_no_scale.bigWig \
    --tail-edge --read-size=30
seqOutBias hg38.fa $mfiles --no-scale --bw=HEK_minus_combined_no_scale.bigWig \
    --tail-edge --read-size=30


mkdir combined_bigWig
mv *combined_no_scale.bigWig combined_bigWig

gzip *not_scaled.bed

#extract all complete gene annotations
awk '$3 == "gene"' Homo_sapiens.GRCh38.87.gtf | \
    sed 's/^/chr/' | \
    awk '{OFS="\t";} {print $1,$4,$5,$14,$10,$7}' | \
    sed 's/";//g' | \
    sed 's/"//g' > Homo_sapiens.GRCh38.87.bed
```

Operating on combined `bigWig` and gene annotation files in `R`. The following chunk loads the functions from our `seqOutBias` paper and a set of functions that are specific for this project.

```
direc = paste0(path.expand("~"),'/ZNF143/')

source('https://raw.githubusercontent.com/guertinlab/seqOutBias/master/docs/R/seqOutBias_functions.R')
source('https://raw.githubusercontent.com/mjg54/znf143_pro_seq_analysis/master/docs/ZNF143_functions.R')

library(DESeq2)
library(lattice)
library(bigWig)
library(grid)
library(zoo)
library(latticeExtra)
library(data.table)

sessionInfo()

## R version 3.4.0 (2017-04-21)
## Platform: x86_64-apple-darwin15.6.0 (64-bit)
```

```
## Running under: macOS  10.13.6
##
## Matrix products: default
## BLAS: /Library/Frameworks/R.framework/Versions/3.4/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/3.4/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
##  [1] grid      parallel  stats4    stats     graphics  grDevices utils
##  [8] datasets  methods   base
##
## other attached packages:
##  [1] knitr_1.20                  data.table_1.11.4
##  [3] latticeExtra_0.6-28         RColorBrewer_1.1-2
##  [5] zoo_1.8-4                   DESeq2_1.16.1
##  [7] SummarizedExperiment_1.6.5 DelayedArray_0.2.7
##  [9] matrixStats_0.53.1          Biobase_2.36.2
## [11] GenomicRanges_1.28.6        GenomeInfoDb_1.12.3
## [13] IRanges_2.10.5              S4Vectors_0.14.7
## [15] BiocGenerics_0.22.1         gtools_3.5.0
## [17] gplots_3.0.1                bigWig_0.2-9
## [19] lattice_0.20-35
##
## loaded via a namespace (and not attached):
##  [1] bit64_0.9-7              splines_3.4.0
##  [3] Formula_1.2-3            highr_0.7
##  [5] blob_1.1.1               GenomeInfoDbData_0.99.0
##  [7] pillar_1.2.3             RSQLite_2.1.1
##  [9] backports_1.1.2          digest_0.6.15
## [11] XVector_0.16.0           checkmate_1.8.5
## [13] colorspace_1.3-2         htmltools_0.3.6
## [15] Matrix_1.2-14            plyr_1.8.4
## [17] XML_3.98-1.11            genefilter_1.58.1
## [19] zlibbioc_1.22.0          xtable_1.8-2
## [21] scales_0.5.0             gdata_2.18.0
## [23] BiocParallel_1.10.1      htmlTable_1.12
## [25] tibble_1.4.2             annotate_1.54.0
## [27] ggplot2_2.2.1            nnet_7.3-12
## [29] lazyeval_0.2.1           survival_2.42-3
## [31] magrittr_1.5             memoise_1.1.0
## [33] evaluate_0.10.1          foreign_0.8-70
## [35] tools_3.4.0              stringr_1.3.1
## [37] munsell_0.5.0            locfit_1.5-9.1
## [39] cluster_2.0.7-1          AnnotationDbi_1.38.2
## [41] compiler_3.4.0           caTools_1.17.1
## [43] rlang_0.2.1              RCurl_1.95-4.10
## [45] rstudioapi_0.7           htmlwidgets_1.2
## [47] bitops_1.0-6             base64enc_0.1-3
## [49] labeling_0.3             gtable_0.2.0
## [51] DBI_1.0.0                gridExtra_2.3
## [53] bit_1.1-14               Hmisc_4.1-1
## [55] KernSmooth_2.23-15       stringi_1.2.3
## [57] Rcpp_0.12.17             geneplotter_1.54.0
## [59] rpart_4.1-13             acepack_1.4.1
```

```
setwd(direc)
options(scipen=999)

#clean up annotations
tss.test = read.table('Homo_sapiens.GRCh38.87.tss.bed')
tss.test.100 = bed.100.interval(tss.test)
tss.test.100[,1] = as.character(tss.test.100[,1])
chr_keep = c( paste0("chr",c(1:22)), "chrX", "chrY")
tss.test.100 = tss.test.100[tss.test.100[,1] %in% chr_keep,]
tss.test.100[tss.test.100[,1]=="chrMT",1] = "chrM"

#load combined bigWigs
loaded.bw.plus = load.bigWig(paste0(direc,
                                    'combined_bigWig/HEK_plus_combined_no_scale.bigWig'))
loaded.bw.minus = load.bigWig(paste0(direc,
                                     'combined_bigWig/HEK_minus_combined_no_scale.bigWig'))

#count in pause region
mod.inten = bed6.region.bpQuery.bigWig(loaded.bw.plus, loaded.bw.minus, tss.test.100)
mod.inten.df = data.frame(cbind(tss.test.100, mod.inten))
mod.inten.df = as.data.table(mod.inten.df)

#select the TSS with highest read count
df.tss = mod.inten.df[mod.inten.df[, .I[which.max(mod.inten)], by=xy]$V1]

save(df.tss, file = paste0(direc, 'df.tss.hek.Rdata'))


#combine with transcription termination site. One per gene
```

```
bed6 = read.table("Homo_sapiens.GRCh38.87.bed")
gene.file = bed6[bed6[,1] %in% chr_keep,]
gene.file[,1] = as.character(gene.file[,1])
gene.file[,6] = as.character(gene.file[,6])
gene.file = gene.file[!duplicated(paste(gene.file[,1],gene.file[,2],gene.file[,3])),]

#ad hoc filtering of small genes and problem annotations
gene.file = gene.file[(gene.file[,3] - gene.file[,2] )> 200,]
gene.file = gene.file[gene.file[,4] != 'Metazoa_SRP' &
                      gene.file[,4] != 'U3' & gene.file[,4] != '7SK',]
gene.file = gene.file[!duplicated(gene.file[,4]),]

#merge
df.tss.pre = merge(gene.file, df.tss, by.x='V4', by.y='xy')[,c(2,3,4,10,1,6,8,9)]

colnames(df.tss.pre) = c('chr', 'start', 'end', 'gene', 'xy', 'strand', 'tssplus', 'tssminus')


df.tss.pre[df.tss.pre$strand == '+',]$start = df.tss.pre[df.tss.pre$strand == '+',]$tssplus - 20
df.tss.pre[df.tss.pre$strand == '-',]$end = df.tss.pre[df.tss.pre$strand == '-',]$tssminus + 20

df.tss.pre = df.tss.pre[,c(1:3,5,4,6)]
colnames(df.tss.pre) = c('V1', 'V2','V3','V4','V5','V6')

gene.file = df.tss.pre
dim(gene.file)
gene.file = gene.file[(gene.file[,3] - gene.file[,2] )> 200,]
gene.file = gene.file[(gene.file[,3] - gene.file[,2] ) < 2500000,]

dim(gene.file)

#I may be missing some weird annotation error that puts
#the 5' end of a gene incredibly far aawy from the true tss
```

1.4 Counting in gene coordinates

Section 1.3 defines each gene annotation for downstream analyses. The following section will analyze the reads that align to these annotations. The principle components analysis should confirm that replicates cluster and treatments separate.

```
df.HEK = get.raw.counts.interval(gene.file, direc, file.prefix = 'HEK_')


colnames(df.HEK) = sapply(strsplit(colnames(df.HEK), '_PE1'), '[', 1)

#use this for making the bedGraphs and loading into UCSC
estimateSizeFactorsForMatrix(df.HEK)
write.table(estimateSizeFactorsForMatrix(df.HEK),
            file = 'norm.bedGraph.sizeFactor.txt', quote =F, col.names=F)

#PCA for experiments

df.HEK.deseq.raw = run.deseq.table(df.HEK)
rld_HEK = rlogTransformation(df.HEK.deseq.raw)
plotPCA(rld_HEK, intgroup="condition")
x = plotPCA(rld_HEK, intgroup="condition", returnData=TRUE)
percentVar = round(100 * attr(x, "percentVar"))

#this function was updated in the ATAC analysis to allow for different # reps
plotPCAlattice(x, file = paste0(direc, 'PCA_HEK_lattice_guertin.pdf'), reps = 4)


df.HEK.deseq.raw.2 = run.deseq.table(df.HEK[,c(1:16)])
rld_HEK.2 = rlogTransformation(df.HEK.deseq.raw.2)
plotPCA(rld_HEK.2, intgroup="condition")
x.2 = plotPCA(rld_HEK.2, intgroup="condition", returnData=TRUE)
percentVar = round(100 * attr(x.2, "percentVar"))

#this function was updated in the ATAC analysis to allow for different # reps
plotPCAlattice(x.2, file = paste0(direc, 'PCA_HEK_ZNF143_tagged_lattice_guertin.pdf'), reps = 4)


save.image(file = paste0(direc, '190212_HEK_image.Rdata'))
```

Conclusion: Despite variation in read depth, the replicates are consistent as represented by PCA.

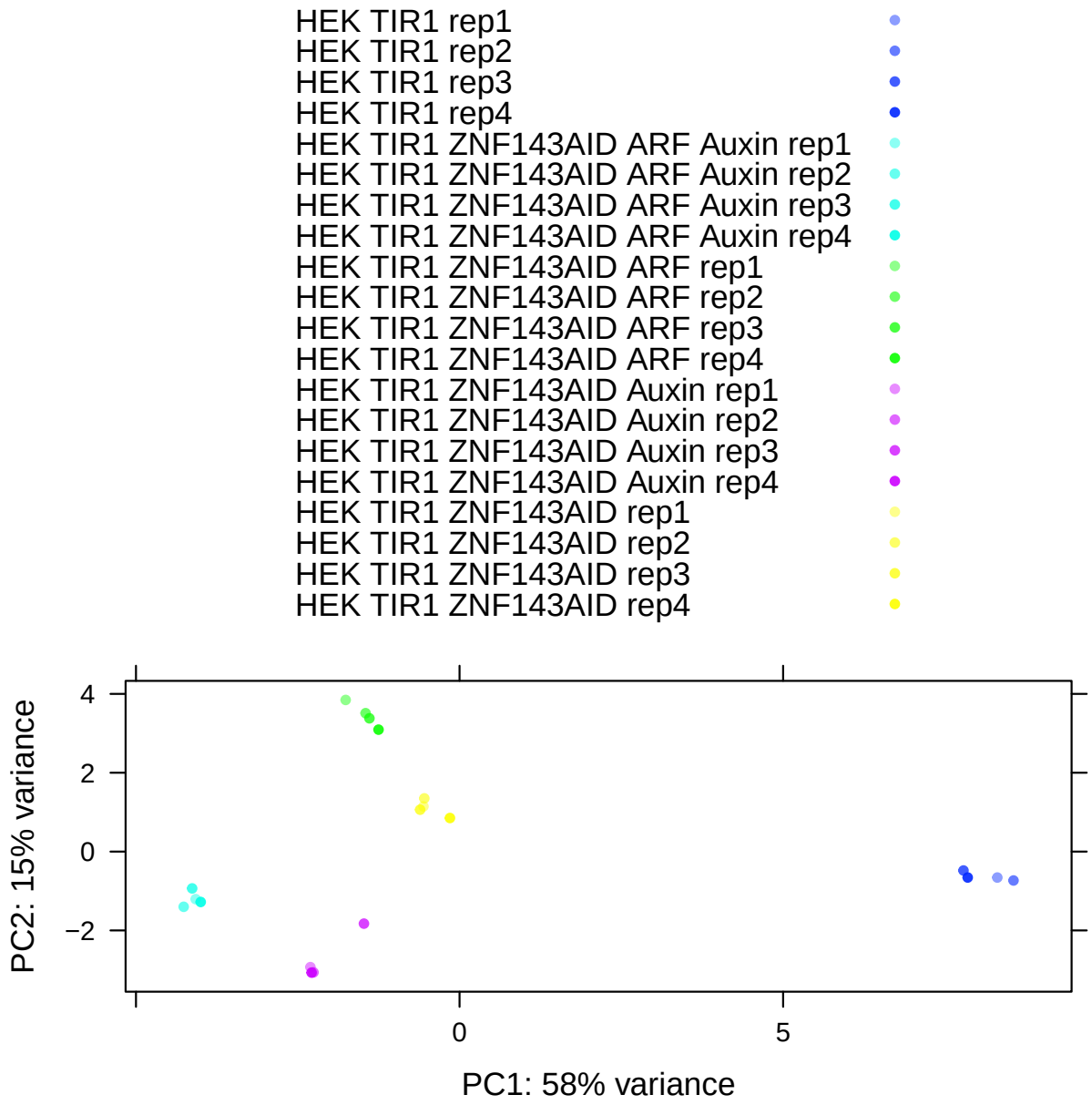

Figure 1: Principle components analysis indicates that ZNF143 tagging dominates PC1 and all replicates cluster together and distinct from other conditions.

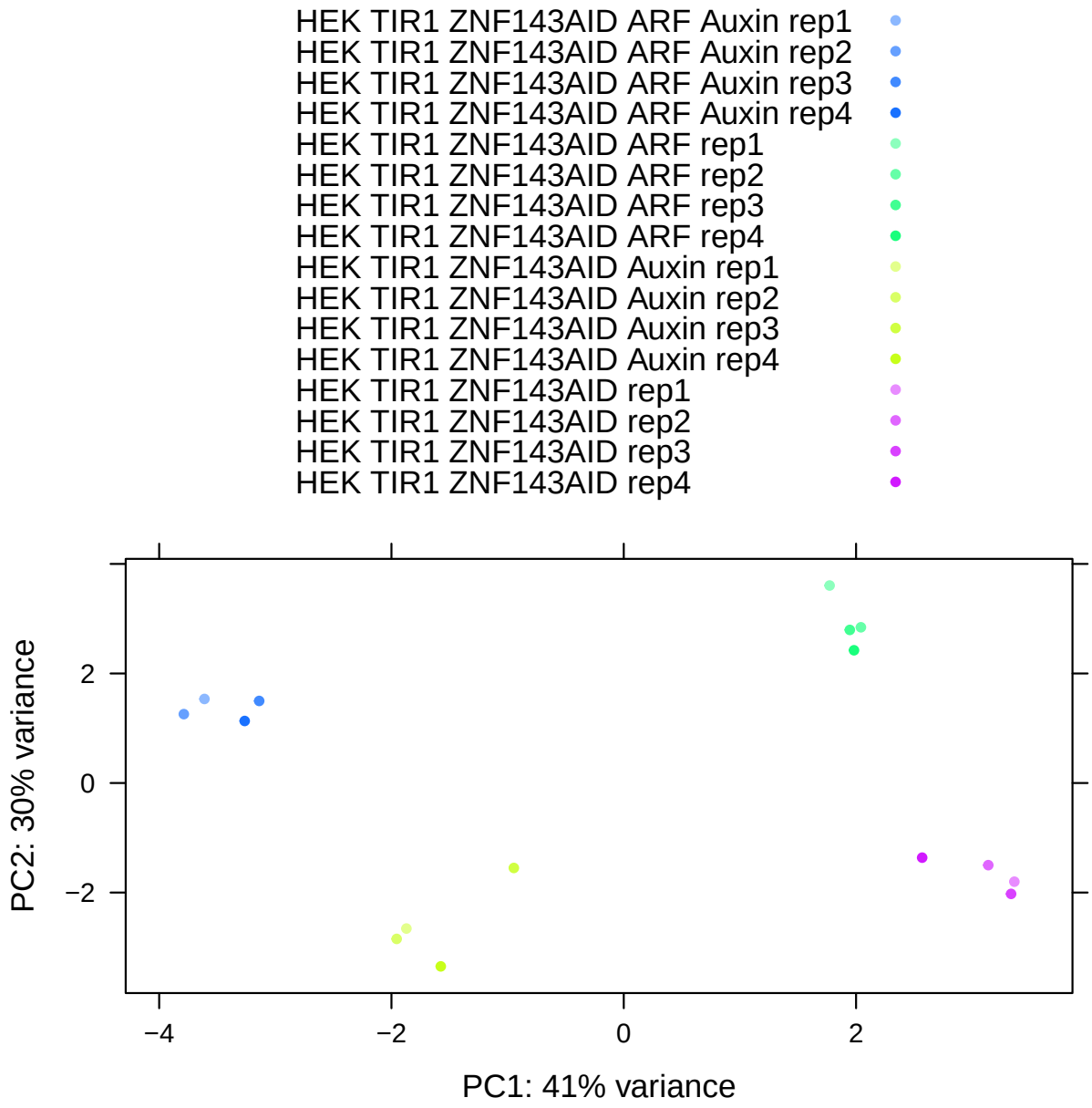

Figure 2: Principle components analysis indicates that auxin treatment dominates PC1 and ARF rescue dominates PC2. All replicates cluster together.

1.5 Replicate correlations

```
#replicate concordance:
rep1.corr = df.HEK[,seq(1, ncol(df.HEK), by = 4)]
rep2.corr = df.HEK[,seq(2, ncol(df.HEK), by = 4)]
rep3.corr = df.HEK[,seq(3, ncol(df.HEK), by = 4)]
rep4.corr = df.HEK[,seq(4, ncol(df.HEK), by = 4)]
colnames(rep1.corr) = gsub('_', ' ', sapply(strsplit(colnames(rep1.corr), '_rep'), '[', 1))
colnames(rep2.corr) = gsub('_', ' ', sapply(strsplit(colnames(rep2.corr), '_rep'), '[', 1))
colnames(rep3.corr) = gsub('_', ' ', sapply(strsplit(colnames(rep3.corr), '_rep'), '[', 1))
colnames(rep4.corr) = gsub('_', ' ', sapply(strsplit(colnames(rep4.corr), '_rep'), '[', 1))

lattice.pairwise.scatter(rep1.corr, rep2.corr,
                         file = paste0(direc, 'Fig_pairwise_scatter_vCAP_reps1v2_lattice.pdf'),
                         r.1 = '1', r.2 ='2')
lattice.pairwise.scatter(rep3.corr, rep4.corr,
                         file = paste0(direc, 'Fig_pairwise_scatter_vCAP_reps3v4_lattice.pdf'),
                         r.1 = '3', r.2 ='4')
#one can repeat the analyses for all pairwise replicate combinations
```

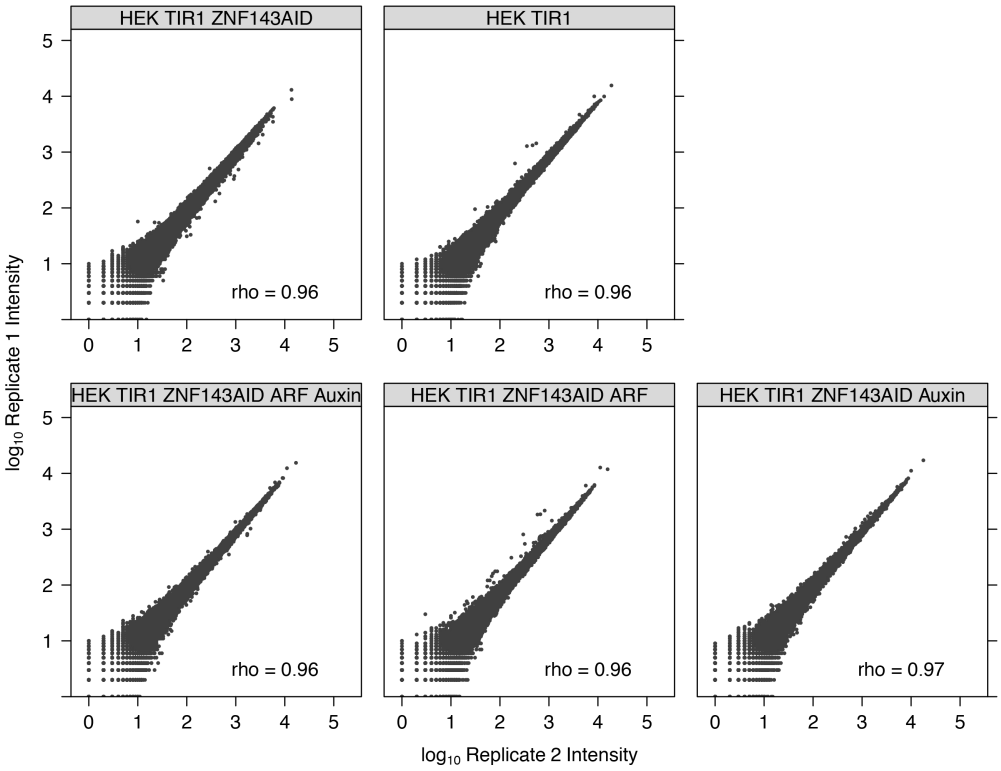

Figure 3: All PRO-seq replicates correlate with Spearman coefficients above 0.9.

2 Aryl Hydrocarbon Receptor regulation–a control experiment

As a control, we treated the progenitor cell line with auxin alone and compared to DMSO. This will identify auxin responsive genes.

```
for i in HEK2*.fastq.gz
do
    name=$(echo $i | awk -F"/" '{print $NF}' | \
           awk -F".fastq.gz" '{print $1}')
    echo $name
    cutadapt -m 26 -a TGGAATTCTCGGGTGCCAAGG ${name}.fastq.gz | \
       fqdedup -i - -o - | \
       fastx_trimmer -f 9 -l 38 | \
       fastx_reverse_complement -z -o ${name}.processed.fastq.gz
    bowtie2 -p 3 -x hg38 -U ${name}.processed.fastq.gz | \
       samtools view -b - | \
       samtools sort - -o ${name}.sorted.bam
       samtools view -bh -F 20 ${name}.sorted.bam > ${name}_pro_plus.bam
       samtools view -bh -f 0x10 ${name}.sorted.bam > ${name}_pro_minus.bam
       seqOutBias hg38.fa ${name}_pro_plus.bam --no-scale --skip-bed \
           --bw=${name}_plus_body_0-mer.bigWig --tail-edge --read-size=30
       seqOutBias hg38.fa ${name}_pro_minus.bam --no-scale --skip-bed \
           --bw=${name}_minus_body_0-mer.bigWig --tail-edge --read-size=30
done
```

```
df.HEK.ctrl = get.raw.counts.interval(gene.file, direc, file.prefix = 'HEK2')


estimateSizeFactorsForMatrix(df.HEK.ctrl)
write.table(estimateSizeFactorsForMatrix(df.HEK.ctrl),
            file = paste0(direc, 'norm.bedGraph.sizeFactor.ctrl.txt'), quote =F, col.names=F)
```

```
wget https://raw.githubusercontent.com/mjg54/znf143_pro_seq_analysis/master/docs/normalize_bedGraph.py
wget https://hgdownload-test.gi.ucsc.edu/goldenPath/hg38/bigZips/hg38.chrom.sizes

for i in HEK2*_rep*_pro_*.bam
do
    name=$(echo $i | awk -F"/" '{print $NF}' | awk -F"_pro_" '{print $1}')
    strand=$(echo $i | awk -F"pro_" '{print $NF}' | awk -F".bam" '{print $1}')
    invscale1=$(grep ${name} norm.bedGraph.sizeFactor.ctrl.txt | awk -F" " '{print $2}')
    invscale=$(echo $invscale1 |bc)
    scaletrue=$(bc <<< "scale=4 ; 1.0 / $invscale")
    echo $name
    echo $strand
    echo $scaletrue
    echo file_to_scale
    echo ${name}_${strand}_body_0-mer.bigWig
    bigWigToBedGraph ${name}_${strand}_body_0-mer.bigWig ${name}_${strand}_body_0-mer.bg
    echo normalizing
    python ~/ZNF143/normalize_bedGraph.py -i ${name}_${strand}_body_0-mer.bg \
       -s $scaletrue -o ${name}_${strand}_scaled.bg
    bedGraphToBigWig ${name}_${strand}_scaled.bg hg38.chrom.sizes ${name}_pro_${strand}_scaled.bigWig
    rm ${name}_${strand}_scaled.bg
    rm ${name}_${strand}_body_0-mer.bg
done
```

2.1 Differential Expression upon Auxin treatment

```
df.dmso.auxin.deseq = run.deseq.list(df.HEK.ctrl[,c(
'HEK293T_TIR1_C14_3hrDMSO_rep1',
'HEK293T_TIR1_C14_3hrDMSO_rep2',
'HEK293T_TIR1_C14_3hrAUXIN_rep1',
'HEK293T_TIR1_C14_3hrAUXIN_rep2')])

df.HEK.auxindmso.deseq.effects.lattice =
    categorize.deseq.df(df.dmso.auxin.deseq,
                        fdr = 0.05, log2fold = 0.0, treat = 'Auxin Alone')

pdf("MA_plot_AuxinDMSO_response_in_classes_panels.pdf", useDingbats = FALSE, width=6.83, height=3.33);
print(xyplot(df.HEK.auxindmso.deseq.effects.lattice$log2FoldChange ~
                log(df.HEK.auxindmso.deseq.effects.lattice$baseMean, base=10),
             groups=df.HEK.auxindmso.deseq.effects.lattice$arfauxin,
             col=c("grey90", "grey60", "red", "blue"),
```

```
              scales="free",
              aspect=1,
              ylim=c(-4.2, 4.2),
              xlim=c(-1,4.2),
              par.strip.text=list(cex=1.0, font = 1),
              pch=20,
              cex=0.5,
              ylab=expression("+Auxin log"[2]~"PRO fold change"),
              xlab=expression("log"[10]~"Mean of Normalized Counts"),
             par.settings=list(par.xlab.text=list(cex=1.1,font=2),
                               par.ylab.text=list(cex=1.1,font=2),
                               strip.background=list(col="grey85"))))
dev.off()

df.dmso.auxin.deseq.activated =
    df.HEK.auxindmso.deseq.effects.lattice[df.HEK.auxindmso.deseq.effects.lattice$arfauxin ==
                                           'Auxin Alone Activated',]

write.table(cbind(do.call(rbind, strsplit(rownames(df.dmso.auxin.deseq.activated),"[-,_,:]")),
df.dmso.auxin.deseq.activated[,2], df.dmso.auxin.deseq.activated[,6])[,4],
row.names=F, col.names=F, quote=F, file = 'dmso.auxin.deseq.activated.txt')
```

We observe very few repressed genes, but over a range of FDR threshlds we consistently observe that the activated gene class is enriched in Aryl Hydrocarbon Receptor position weight matrices in the promoter and the most enriched pathway is the Aryl Hydrocarbon Receptor pathway (q-value = 0.002) (Kuleshov *et al.*, 2016). This led us to ask whether activated genes are enriched proximal to AHR binding sites.

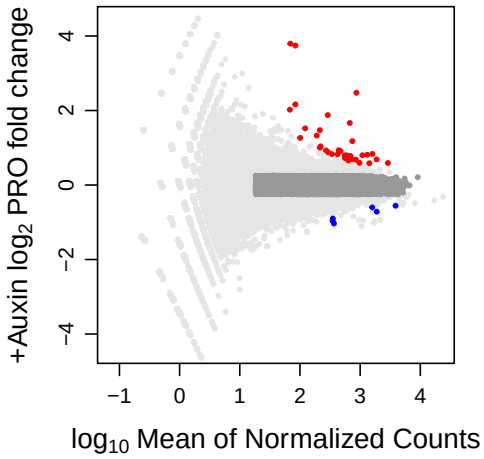

Figure 4: Genes are activated by Auxin alone are enriched for AHR motifs and the genes are within AHR pathways.

2.2 AHR ChIP-seq and proximity analysis

ChIP-seq analysis of AHR binding. Call peaks with MACS2 (Zhang *et al.*, 2008) using the raw data from (Lo and Matthews, 2012).

```
#AHR_TCDD_45min_ChIPSeq
fastq-dump SRR5057969
fastq-dump SRR5057970
fastq-dump SRR5057971

#AHR_DMSO_45min_ChIPSeq
fastq-dump SRR5057966
fastq-dump SRR5057967
fastq-dump SRR5057968

gzip *fastq

cat SRR5057969.fastq.gz SRR5057970.fastq.gz \
SRR5057971.fastq.gz > AHR_TCDD_45min_MCF7_chip.fastq.gz
cat SRR5057966.fastq.gz SRR5057967.fastq.gz \
SRR5057968.fastq.gz > AHR_DMSO_45min_MCF7_chip.fastq.gz

rm SRR*gz

for fq in AHR*.fastq.gz
do
    name=$(echo $fq | awk -F"/" '{print $NF}' | awk -F".fastq.gz" '{print $1}')
    echo $name
    bowtie2 -p 3 -x hg38 -U ${fq} | \
       samtools view -b - | \
       samtools sort - -o ${name}.sorted.bam
done


ctrl=AHR_DMSO_45min_MCF7_chip.sorted.bam
for i in AHR_TCD*chip*sorted.bam
do
    name=$(echo $i | awk -F"/" '{print $NF}' | awk -F".sorted.bam" '{print $1}')
    echo $name
    macs2 callpeak -t $i -c $ctrl -f BAM --keep-dup 10 -n $name -g hs -B -m 10 100
done
```

```
ahr.peaks = read.table(paste0(direc, 'AHR_TCDD_45min_MCF7_chip_summits.bed'),
                       sep = '\t', header =FALSE)

colnames(df.HEK.auxindmso.deseq.effects.lattice) =
    c(colnames(df.HEK.auxindmso.deseq.effects.lattice)[c(1:
    (length(colnames(df.HEK.auxindmso.deseq.effects.lattice))-1))],
      'arfauxin')
df.all.ahr = cdf.deseq.df(df = df.HEK.auxindmso.deseq.effects.lattice,
            genes = gene.file, chip.peaks ='AHR_TCDD_45min_MCF7_chip_summits.bed',
            treat = 'Auxin Alone', tf.name = 'Aryl Hydrocarbon Receptor')


pdf(paste0(direc, "Figure_cdf_compare_Reg_classes_AHR.pdf"), width=4.2, height=3.83)

        col.lines = c("#FF0000" , "grey60", "#0000FF","grey90")
        ecdfplot(~log(abs(dis), base = 10), groups = status, data = df.all.ahr,
                 auto.key = list(lines=TRUE, points=FALSE),
                 col = col.lines,
                 aspect = 1,
                 scales=list(relation="free",alternating=c(1,1,1,1)),
                 ylab = 'Cumulative Distribution Function',
                 xlab = expression('log'[10]~'AHR Distance from TSS'),
                 between=list(y=1.0),
                 type = 'a',
                 xlim = c(0,5.5),
                 lwd=2,
                 par.settings = list(superpose.line = list(col = col.lines, lwd=3),
                                     strip.background=list(col="grey85")),
                 panel = function(...) {
                     panel.abline(v= 200, lty =2)
                     panel.ecdfplot(...)
                 })
dev.off()


ks.auxindmso.ActvUnc = ks.test(x = log(abs(df.all.ahr$dis)[df.all.ahr$status == 'Auxin Alone Activated']),
```

```
        y = log(abs(df.all.ahr$dis)[df.all.ahr$status == 'Auxin Alone Unchanged']))$p.value
ks.auxindmso.ActvUnc
ks.auxindmso.ActvUnc = formatC(ks.auxindmso.ActvUnc, format = "e", digits = 3)
ks.auxindmso.RepvUnc = ks.test(x = log(abs(df.all.ahr$dis)[df.all.ahr$status == 'Auxin Alone Repressed']),
        y = log(abs(df.all.ahr$dis)[df.all.ahr$status == 'Auxin Alone Unchanged']))$p.value
ks.auxindmso.RepvUnc

save(ks.auxindmso.ActvUnc, ks.auxindmso.RepvUnc, file = paste0(direc, 'ks.auxindmso.Rdata'))
```

We find that AHR binding, as measured by ChIP-seq, is exclusively enriched proximal to the activated gene class (Kolmogorov–Smirnov two-sided p-value = $1.129e-10$) and not the repressed gene class (Kolmogorov–Smirnov two-sided p-value = 0.7511954) .

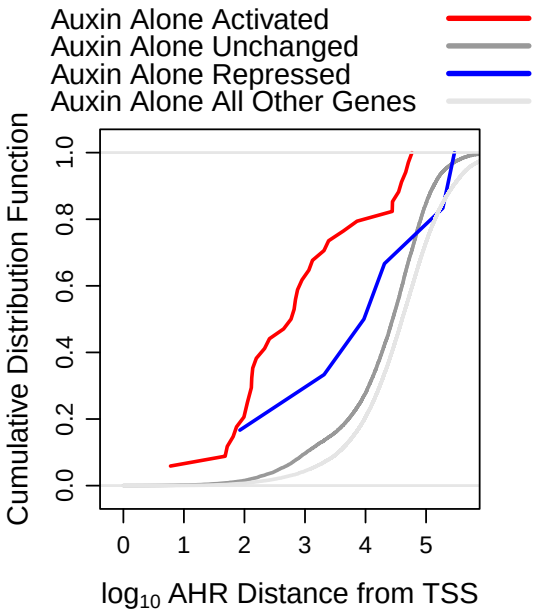

Figure 5: Genes are activated in the presence of Auxin are proximal to AHR binding sites.

3 ZNF143 depletion analyses

We will first explore how auxin treatment affects gene expression in the chronic ZNF143 depletion cells and the ARF rescue cells. We will next determine the proximity of differentially expressed gene classes to ZNF143 binding sites.

3.1 Chronic depletion analysis

We will employ `DESeq2` (Love *et al.*, 2014) to identify differentially expressed genes in the chronic ZNF143 depleted background and we investigate the degree to which ARF rescue changes the expression of these genes.

```
fdr = 0.001
log2fold = 0.0

df.HEK.chronic.deseq = run.deseq.list.any(df.HEK[,c(
    'HEK_TIR1_rep1',
    'HEK_TIR1_rep2',
    'HEK_TIR1_rep3',
    'HEK_TIR1_rep4',
    'HEK_TIR1_ZNF143AID_rep1',
    'HEK_TIR1_ZNF143AID_rep2',
    'HEK_TIR1_ZNF143AID_rep3',
    'HEK_TIR1_ZNF143AID_rep4')], unt = 4, trt = 4)

plot.ma.lattice(df.HEK.chronic.deseq,
                filename = paste('df.HEK.chronic.deseq', 'FDR', fdr, 'log2', log2fold, sep =''),
                p=fdr, title.main = 'Differential PRO',log2fold = log2fold)
```

```
categorize.deseq.df.mods <- function(df, fdr = 0.05, log2fold = 0.0, treat = 'Auxin') {
    df.activated = df[df$padj < fdr & !is.na(df$padj) & df$log2FoldChange > log2fold,]

    df.repressed = df[df$padj < fdr & !is.na(df$padj) & df$log2FoldChange < -log2fold,]

    df.unchanged = df[df$padj > 0.5 & !is.na(df$padj) & abs(df$log2FoldChange) < 0.25,]

    df.dregs = df[!(df$padj < fdr & !is.na(df$padj) & df$log2FoldChange > log2fold) &
                  !(df$padj < fdr & !is.na(df$padj) & df$log2FoldChange < -log2fold) &
                  !(df$padj > 0.5 & !is.na(df$padj) & abs(df$log2FoldChange) < 0.25), ]
    df.unchanged$arfauxin = paste(treat, 'Unchanged')
    df.activated$arfauxin = paste(treat, 'Activated')
    df.repressed$arfauxin = paste(treat, 'Repressed')
    df.dregs$arfauxin = paste(treat, 'All Other Genes')
    df.effects.lattice =
    rbind(df.activated,
          df.unchanged,
          df.repressed,
          df.dregs)
    df.effects.lattice[grep('ZNF143', rownames(df.effects.lattice)),]$arfauxin = 'AAA'
    df.effects.lattice$arfauxin = factor(df.effects.lattice$arfauxin)
    df.effects.lattice$arfauxin = relevel(df.effects.lattice$arfauxin, ref = "AAA")
    df.effects.lattice$arfauxin = relevel(df.effects.lattice$arfauxin, ref = paste(treat, 'Activated'))
    df.effects.lattice$arfauxin = relevel(df.effects.lattice$arfauxin, ref = paste(treat, 'Repressed'))
    df.effects.lattice$arfauxin = relevel(df.effects.lattice$arfauxin, ref = paste(treat, 'Unchanged'))
    df.effects.lattice$arfauxin = relevel(df.effects.lattice$arfauxin, ref = paste(treat, 'All Other Genes'))
    return(df.effects.lattice)
}

df.HEK.chronic.deseq.lattice =
    categorize.deseq.df.mods(df.HEK.chronic.deseq, fdr = 0.001, log2fold = 0.0, treat = 'Chronic Depletion')


pdf(paste0(direc, "MA_plot_Chronic_depletion_in_classes.pdf"), useDingbats = FALSE, width=3.83, height=3.33);
print(xyplot(df.HEK.chronic.deseq.lattice$log2FoldChange ~
                log(df.HEK.chronic.deseq.lattice$baseMean, base=10),
             groups=df.HEK.chronic.deseq.lattice$arfauxin,
             col=c("grey80", "grey55", "#0000FF", "#FF0000", "#000000"),
             #main='ZNF143 Degradation Classes of Genes',
             scales="free",
             aspect=1,
             ylim=c(-4.2, 4.2),
             xlim=c(-1,4.2),
             par.strip.text=list(cex=1.0, font = 1),
             pch=20,
             cex=c(rep(0.5, 4), 1),
             ylab=expression("AID tag log"[2]~"PRO fold change"),
             xlab=expression("log"[10]~"Mean of Normalized Counts"),
            par.settings=list(par.xlab.text=list(cex=1.1,font=2),
                              par.ylab.text=list(cex=1.1,font=2),
                              strip.background=list(col="grey85"))))
dev.off()


df.HEK.ZNF143.ArfRescue.deseq = run.deseq.list.any(df.HEK[,c(
    'HEK_TIR1_ZNF143AID_rep1',
    'HEK_TIR1_ZNF143AID_rep2',
    'HEK_TIR1_ZNF143AID_rep3',
    'HEK_TIR1_ZNF143AID_rep4',
    'HEK_TIR1_ZNF143AID_ARF_rep1',
    'HEK_TIR1_ZNF143AID_ARF_rep2',
    'HEK_TIR1_ZNF143AID_ARF_rep3',
    'HEK_TIR1_ZNF143AID_ARF_rep4')], unt = 4, trt = 4)

plot.ma.lattice(df.HEK.ZNF143.ArfRescue.deseq,
                filename = paste('df.HEK.ZNF143.ArfRescue.deseq', 'FDR', fdr, 'log2', log2fold, sep =''),
                p=fdr, title.main = 'Differential PRO',log2fold = log2fold)
df.HEK.arf.rescue.deseq.effects.lattice =
    categorize.deseq.df(df.HEK.ZNF143.ArfRescue.deseq, fdr = 0.001, log2fold = 0.0, treat = 'Auxin')

df.HEK.arf.rescue.deseq.effects.lattice[, ncol(df.HEK.arf.rescue.deseq.effects.lattice) + 1] = 'ARF Response'
colnames(df.HEK.arf.rescue.deseq.effects.lattice) =
    c(colnames(df.HEK.arf.rescue.deseq.effects.lattice)[1:ncol(df.HEK.arf.rescue.deseq.effects.lattice) -1],
      'rescue')


df.chronic.effects = merge(df.HEK.chronic.deseq.lattice, df.HEK.arf.rescue.deseq.effects.lattice,
                           by="row.names")

rownames(df.chronic.effects) = df.chronic.effects[,1]
colnames(df.chronic.effects) = c("rownames", "baseMean.chronic", "log2FoldChange.chronic",
                                 "lfcSE.chronic", "stat.chronic", "pvalue.chronic",
                                "padj.chronic",  "chronic", "baseMean.rescue",
                                "log2FoldChange.rescue", "lfcSE.rescue", "stat.rescue",
                                "pvalue.rescue", "padj.rescue","arfrescue","rescue")
```

```
chronic.repressed = length(df.chronic.effects$chronic[df.chronic.effects$chronic ==
                                                      'Chronic Depletion Repressed'])
chronic.activated = length(df.chronic.effects$chronic[df.chronic.effects$chronic ==
                                                      'Chronic Depletion Activated'])

chronic.repressed.act = length(df.chronic.effects$log2FoldChange.rescue[df.chronic.effects$chronic ==
                                  'Chronic Depletion Repressed' &
                                  df.chronic.effects$log2FoldChange.rescue > 0])

chronic.repressed.rep = length(df.chronic.effects$log2FoldChange.rescue[df.chronic.effects$chronic ==
                                  'Chronic Depletion Repressed'&
                                  df.chronic.effects$log2FoldChange.rescue < 0])

chronic.activated.act = length(df.chronic.effects$log2FoldChange.chronic[df.chronic.effects$chronic ==
                                  'Chronic Depletion Activated' &
                                  df.chronic.effects$log2FoldChange.rescue > 0])

chronic.activated.rep = length(df.chronic.effects$log2FoldChange.chronic[df.chronic.effects$chronic ==
                                  'Chronic Depletion Activated'&
                                  df.chronic.effects$log2FoldChange.rescue < 0])

save(chronic.repressed, chronic.activated,
     chronic.repressed.act, chronic.repressed.rep,
     chronic.activated.act, chronic.activated.rep, file = paste0(direc, 'chronic.numbers.Rdata'))


df.chronic.effects$chronic <- factor(as.character(df.chronic.effects$chronic),
                             levels = c("Chronic Depletion Activated",
                                        "Chronic Depletion Unchanged",
                                        "Chronic Depletion Repressed",
                                        "Chronic Depletion All Other Genes"))


pdf(paste0(direc, 'ARF_Rescue_changes_in_Chronic_classes_bwplot.pdf'),
    useDingbats = FALSE, width=4.4, height=5.0)

col.order = c("#FF0000","grey55", "#0000FF", "grey80" )
trellis.par.set(box.umbrella = list(lty = 1, col=col.order, lwd=1.5),
                box.rectangle = list(col = col.order, lwd=1.5),
                plot.symbol = list(col=c("black"), lwd=1.0, pch = 19, cex = 0.5))

bwplot(df.chronic.effects$log2FoldChange.rescue ~ df.chronic.effects$chronic,
       horizontal =FALSE,
       between=list(y=1.0, x = 1.0),
       scales=list(relation="free",rot = 30, alternating=c(1,1,1,1),
                   x = list(col = col.order)),
       ylab = expression('log'[2]*' (change upon ARF Rescue)'),
       xlab = expression('Chronic Depletion Response Category'),
        pch = '|', col= 'black',
       aspect = 1.75,
       do.out = FALSE,
       par.settings=list(par.xlab.text=list(cex=1.0,font=1),
           par.ylab.text=list(cex=1.0,font=1),
           par.main.text=list(cex=1.0, font=1)),
       strip = function(..., which.panel, bg) {
           bg.col = c("#ce228e" , "#2290cf","grey90","grey60")
           strip.default(..., which.panel = which.panel,
                         bg = rep(bg.col, length = which.panel)[which.panel])
       },
       panel = function(..., box.ratio, col, pch) {
           panel.abline(h = 0, , lty =1, col = 'grey70')
           #panel.violin(..., col = 'white',
           #             varwidth = FALSE, box.ratio = box.ratio)
           panel.stripplot(..., col='grey20', do.out=FALSE, jitter.data=TRUE,
                           pch = 20, cex = 0.12)
           panel.bwplot(..., col = col.order, pch = '|')

       })
dev.off()
```

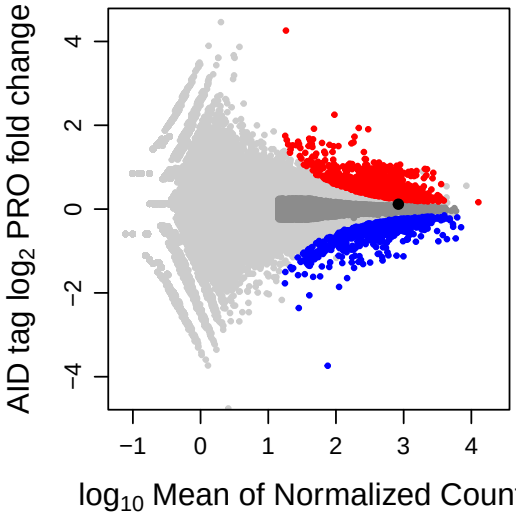

Figure 6: Red genes are activated upon chronic ZNF143 depletion; blue genes are repressed.

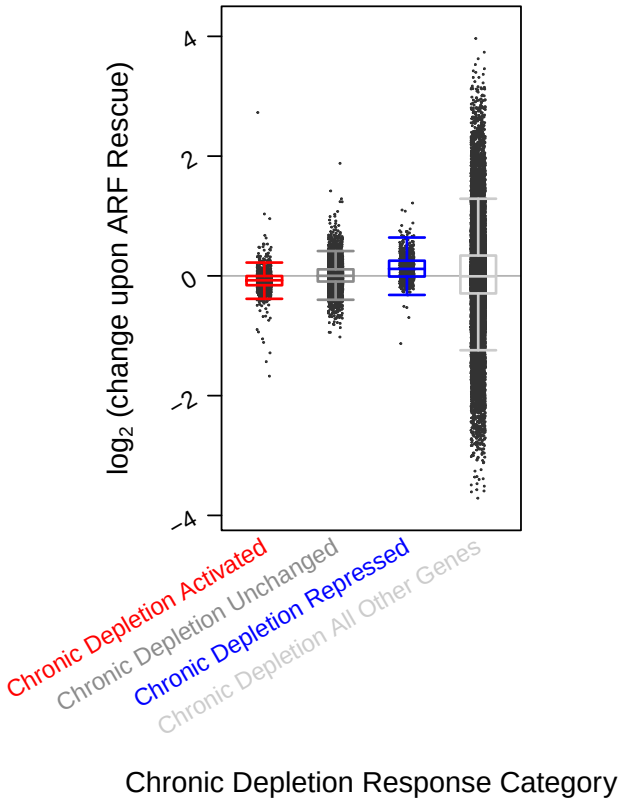

Figure 7: The chronic depletion repressed genes are, on average, activated upon ARF rescue. The chronic depleted activated class are, on average, repressed upon ARF rescue.

Next, we looked at the raw changes in expression upon ARF rescue to determine whether the rescue of ZNF143 stability translated to functional transcriptional rescue genes that change expression when ZNF143 is chronicly depleted. Of the 774 chronic repressed genes, 561 increase their expression upon ARF rescue (Figure 7). Of the 1188 chronic activated genes, 899 decrease their expression upon ARF rescue (Figure 7). These changes are consistent with a functional rescue of gene expression upon ARF rescue of ZNF143 stability.

3.2 Auxin effect in the Chronic Depletion and Rescue backgrounds

We will employ `DESeq2` (Love *et al.*, 2014) to identify differentially expressed genes upon adding auxin to genetically modified HEK-293T cells.

```
fdr = 0.001
log2fold = 0.0

df.HEK.ZNF143.AuxinArf.deseq = run.deseq.list.any(df.HEK[,c(
    'HEK_TIR1_ZNF143AID_ARF_rep1',
    'HEK_TIR1_ZNF143AID_ARF_rep2',
    'HEK_TIR1_ZNF143AID_ARF_rep3',
    'HEK_TIR1_ZNF143AID_ARF_rep4',
    'HEK_TIR1_ZNF143AID_ARF_Auxin_rep1',
    'HEK_TIR1_ZNF143AID_ARF_Auxin_rep2',
    'HEK_TIR1_ZNF143AID_ARF_Auxin_rep3',
    'HEK_TIR1_ZNF143AID_ARF_Auxin_rep4')], unt = 4, trt = 4)

df.HEK.ZNF143.Auxin.deseq = run.deseq.list.any(df.HEK[,c(
    'HEK_TIR1_ZNF143AID_rep1',
    'HEK_TIR1_ZNF143AID_rep2',
    'HEK_TIR1_ZNF143AID_rep3',
    'HEK_TIR1_ZNF143AID_rep4',
    'HEK_TIR1_ZNF143AID_Auxin_rep1',
    'HEK_TIR1_ZNF143AID_Auxin_rep2',
    'HEK_TIR1_ZNF143AID_Auxin_rep3',#
    'HEK_TIR1_ZNF143AID_Auxin_rep4')], unt = 4, trt = 4)


plot.ma.lattice(df.HEK.ZNF143.AuxinArf.deseq,
                filename = paste('df.HEK.ZNF143.AuxinArf.deseq', 'FDR', fdr, 'log2', log2fold, sep =''),
                p=fdr, title.main = 'Differential PRO',log2fold = log2fold)

plot.ma.lattice(df.HEK.ZNF143.Auxin.deseq,
                filename = paste('df.HEK.ZNF143.Auxin.deseq', 'FDR', fdr, 'log2', log2fold, sep =''),
                p=fdr, title.main = 'Differential PRO',log2fold = log2fold)



df.HEK.auxin.arf.deseq.effects.lattice =
    categorize.deseq.df(df.HEK.ZNF143.AuxinArf.deseq, fdr = 0.001, log2fold = 0.0, treat = 'Auxin ARF')
df.HEK.auxin.deseq.effects.lattice =
    categorize.deseq.df(df.HEK.ZNF143.Auxin.deseq, fdr = 0.001, log2fold = 0.0, treat = 'Auxin')


rep.genes =
    sapply(strsplit(rownames(df.HEK.auxin.arf.deseq.effects.lattice[df.HEK.auxin.arf.deseq.effects.lattice$arfauxin ==
                                                                    'Auxin ARF Repressed',]), '_'), '[', 2)


write.table(rep.genes,
            file = "df.HEK.auxin.arf.deseq.effects.lattice_Repressed.txt",
            quote=FALSE, row.names=FALSE, col.names=FALSE)


df.HEK.auxin.arf.deseq.effects.lattice[, ncol(df.HEK.auxin.arf.deseq.effects.lattice) + 1] = 'ARF Rescue'
df.HEK.auxin.deseq.effects.lattice[, ncol(df.HEK.auxin.deseq.effects.lattice) + 1] = 'Chronic Depletion'



colnames(df.HEK.auxin.arf.deseq.effects.lattice) =
    c(colnames(df.HEK.auxin.arf.deseq.effects.lattice)[1:ncol(df.HEK.auxin.arf.deseq.effects.lattice) -1],
      'rescue')


colnames(df.HEK.auxin.deseq.effects.lattice) =
    c(colnames(df.HEK.auxin.deseq.effects.lattice)[1:ncol(df.HEK.auxin.deseq.effects.lattice) -1],
      'rescue')


df.HEK.all.deseq.effects.lattice = rbind(df.HEK.auxin.arf.deseq.effects.lattice,
                                         df.HEK.auxin.deseq.effects.lattice)


pdf("MA_plot_AuxinArf_response_in_classes_panels.pdf", useDingbats = FALSE, width=6.83, height=3.33);
```

```
print(xyplot(df.HEK.all.deseq.effects.lattice$log2FoldChange ~
                log(df.HEK.all.deseq.effects.lattice$baseMean, base=10) |
                df.HEK.all.deseq.effects.lattice$rescue,
             groups=df.HEK.all.deseq.effects.lattice$arfauxin,
             col=c("grey90", "grey60", "#521e77", "#36771e", "grey80", "grey50", "#D60909", "#0931D6"),
             scales="free",
             aspect=1,
             ylim=c(-4.2, 4.2),
             xlim=c(-1,4.2),
             par.strip.text=list(cex=1.0, font = 1),
             pch=20,
             cex=0.5,
             ylab=expression("+Auxin log"[2]~"PRO fold change"),
             xlab=expression("log"[10]~"Mean of Normalized Counts"),
            par.settings=list(par.xlab.text=list(cex=1.1,font=2),
                              par.ylab.text=list(cex=1.1,font=2),
                              strip.background=list(col="grey85"))))
dev.off()

save(df.HEK.all.deseq.effects.lattice,
     file = "df.HEK.all.deseq.effects.lattice.Rdata")
```

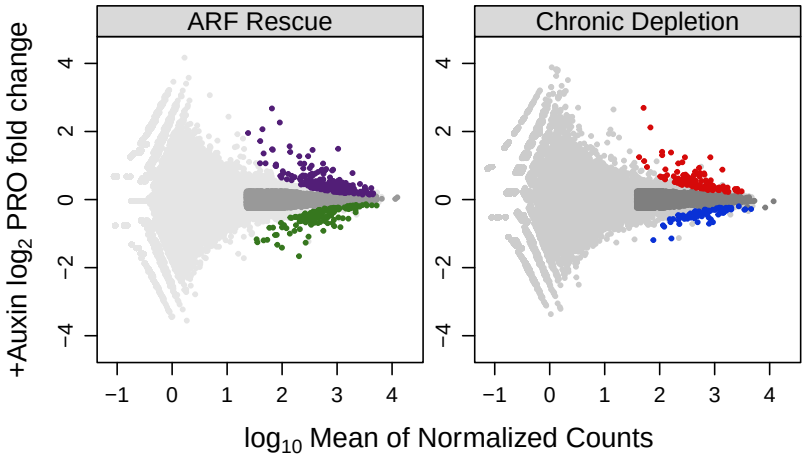

Figure 8: Genes are activated and repressed in the presence of Auxin in both the chronic ZNF143 depleted background and upon ARF rescue.

4 Differential bidirectional transcription at regulatory elements (dREG)

We used dREG to identify regulatory elements by a bidirectional transcription signature (Wang *et al.*, 2018). We used `HEK_plus_combined_no_scale.bigWig` and `HEK_minus_combined_no_scale.bigWig` as the input for dREG. The first step is to retrieve the downstream and upstream coordinates for each element summit so we can count using the appropriate `bigWig` file and look for differential bidirectional transcription upon auxin treatment from the `*.dREG.peak.full.bed` output file.

```
#dREG input
#HEK_plus_combined_no_scale.bigWig
#HEK_minus_combined_no_scale.bigWig

awk '{OFS="\t";} {print $1,$2,$6}' HEK_ZNF143_tag_auxin.dREG.peak.full.bed > \
    HEK_ZNF143_tag_auxin.dREG.peak.minus.bed
awk '{OFS="\t";} {print $1,$6,$3}' HEK_ZNF143_tag_auxin.dREG.peak.full.bed > \
    HEK_ZNF143_tag_auxin.dREG.peak.plus.bed
```

Next we will count the upstream and downstream reads to compare the auxin treatment condition. We are extending the range by 11 base pairs on each side because dREG returns instances where the summit falls outside the range of the peak interval by 10bp, which causes problems in the `bed` file. We also use a very permissive FDR of 0.1 because these regions are short compared to genic annotations and sensitivity to detect changes is reduced. We prefer to sacrifice specificity for sensitivity in downstream motif analysis.

```
get.raw.counts.dREG <- function(df.prefix, path.to.bigWig, file.prefix = 'H') {
    vec.names = c()
    df.plus = read.table(paste0(df.prefix, '.plus.bed'))
    #dREG gives summits outside range for some reason
    df.plus[,2] = df.plus[,2] - 11
    df.plus[,2][df.plus[,2] < 0] = 1
    df.minus = read.table(paste0(df.prefix, '.minus.bed'))
    df.minus[,3] = df.minus[,3] + 11
    inten.df=data.frame(matrix(ncol = 0, nrow = nrow(df.plus)))
    for (mod.bigWig in Sys.glob(file.path(path.to.bigWig,
                        paste(file.prefix, "*plus_body_0-mer.bigWig", sep =''')))) {
        factor.name = strsplit(strsplit(mod.bigWig,
                          "/")[[1]][length(strsplit(mod.bigWig, "/")[[1]])], '_plus')[[1]][1]
        print(factor.name)
        vec.names = c(vec.names, factor.name)
        loaded.bw.plus = load.bigWig(mod.bigWig)
        print(mod.bigWig)
        print(paste(path.to.bigWig,'/',factor.name, '_minus_body_0-mer.bigWig', sep=''))
        loaded.bw.minus = load.bigWig(paste(path.to.bigWig,'/',factor.name,
                          '_minus_body_0-mer.bigWig', sep=''))
        mod.inten.plus = bed.region.bpQuery.bigWig(loaded.bw.plus, df.plus)
        mod.inten.minus = bed.region.bpQuery.bigWig(loaded.bw.minus, df.minus)
        inten.df = cbind(inten.df, mod.inten.plus + mod.inten.minus)
    }
    colnames(inten.df) = vec.names
    r.names = paste(df.plus[,1], ':', df.plus[,2], '-', df.plus[,2] +1, sep='')
    row.names(inten.df) = r.names
    return(inten.df)
}

counts.dreg = get.raw.counts.dREG('HEK_ZNF143_tag_auxin.dREG.peak',
                                  direc, file.prefix = 'HEK_')

colnames(counts.dreg) = sapply(strsplit(colnames(counts.dreg), '_PE1'), '[', 1)

save(counts.dreg, file = 'counts.dreg.Rdata')

fdr = 0.1
log2fold = 0.0

df.HEK.ZNF143.AuxinArf.dREG.deseq = run.deseq.list.any(counts.dreg[,c(
    'HEK_TIR1_ZNF143AID_ARF_rep1',
    'HEK_TIR1_ZNF143AID_ARF_rep2',
    'HEK_TIR1_ZNF143AID_ARF_rep3',
    'HEK_TIR1_ZNF143AID_ARF_rep4',
    'HEK_TIR1_ZNF143AID_ARF_Auxin_rep1',
    'HEK_TIR1_ZNF143AID_ARF_Auxin_rep2',
    'HEK_TIR1_ZNF143AID_ARF_Auxin_rep3',
    'HEK_TIR1_ZNF143AID_ARF_Auxin_rep4')], unt = 4, trt = 4)

plot.ma.lattice(df.HEK.ZNF143.AuxinArf.dREG.deseq,
                filename = paste('df.HEK.ZNF143.AuxinArf.dREG.deseq',
                                 'FDR', fdr, 'log2', log2fold, sep =''),
                p=fdr, title.main = 'Differential PRO',log2fold = log2fold)
```

```
df.HEK.auxin.arf.deseq.effects.dREG.lattice =
    categorize.deseq.df(df.HEK.ZNF143.AuxinArf.dREG.deseq, fdr = 0.1,
                        log2fold = 0.0, treat = 'Auxin ARF')



pdf("MA_plot_AuxinArf_response_in_classes_dREG_panels2.pdf",
    useDingbats = FALSE, width=3.83, height=4.33);
print(xyplot(df.HEK.auxin.arf.deseq.effects.dREG.lattice$log2FoldChange ~
                log(df.HEK.auxin.arf.deseq.effects.dREG.lattice$baseMean, base=10),
             groups=df.HEK.auxin.arf.deseq.effects.dREG.lattice$arfauxin,
             col=c("grey90", "grey60", "#521e77", "#36771e"),
             scales="free",
             aspect=1,
             ylim=c(-4.2, 4.2),
             xlim=c(-1,4.2),
             par.strip.text=list(cex=1.0, font = 1),
             pch=20,
             cex=0.5,
             ylab=expression("+Auxin log"[2]~"PRO fold change at dREG"),
             xlab=expression("log"[10]~"Mean of Normalized Counts"),
            par.settings=list(par.xlab.text=list(cex=1.1,font=2),
                              par.ylab.text=list(cex=1.1,font=2),
                              strip.background=list(col="grey85"))))
dev.off()



process.deseq.df <- function(df, filename = 'file.name', fdr = 0.05, log2fold = 0.0) {
    increased = df[df$padj < fdr & !is.na(df$padj) & df$log2FoldChange > log2fold,]

    decreased = df[df$padj < fdr & !is.na(df$padj) & df$log2FoldChange < -log2fold,]

    unchanged = df[df$padj > 0.5 & !is.na(df$padj) & abs(df$log2FoldChange) < 0.25,]

    dregs = df[!(df$padj < fdr & !is.na(df$padj) & df$log2FoldChange > log2fold) &
                  !(df$padj < fdr & !is.na(df$padj) & df$log2FoldChange < -log2fold) &
                  !(df$padj > 0.5 & !is.na(df$padj) & abs(df$log2FoldChange) < 0.25), ]
    lst = list(increased, decreased, unchanged, dregs)
    process.rows <- function(i, str) {
        coor = rownames(i)
    coor.start = sapply(strsplit(sapply(strsplit(as.character(coor),':'), "[", 2), "-"), "[", 1);
    coor.end = sapply(strsplit(as.character(coor),'-'), "[", 2)
    coor.chr = sapply(strsplit(as.character(coor),':'), "[", 1)
    df.coor = cbind(coor.chr, coor.start, coor.end, as.character(i$baseMean),
                    as.character(i$log2FoldChange), '+', as.character(i$lfcSE),
                    as.character(i$pvalue), as.character(i$padj))
    write.table(df.coor, file = paste(str,'_', filename, '_', fdr, '_FDR.bed', sep =''),
                    sep = '\t', quote=F, row.names=F, col.names=F)
}
    process.rows(increased, 'increased')
    process.rows(decreased, 'decreased')
    process.rows(unchanged, 'unchanged')
    process.rows(dregs, 'dregs')
}

process.deseq.df(df.HEK.auxin.arf.deseq.effects.dREG.lattice,
                 filename = 'df.HEK.auxin.arf.deseq.effects.dREG.lattice', fdr = 0.1, log2fold = 0)
```

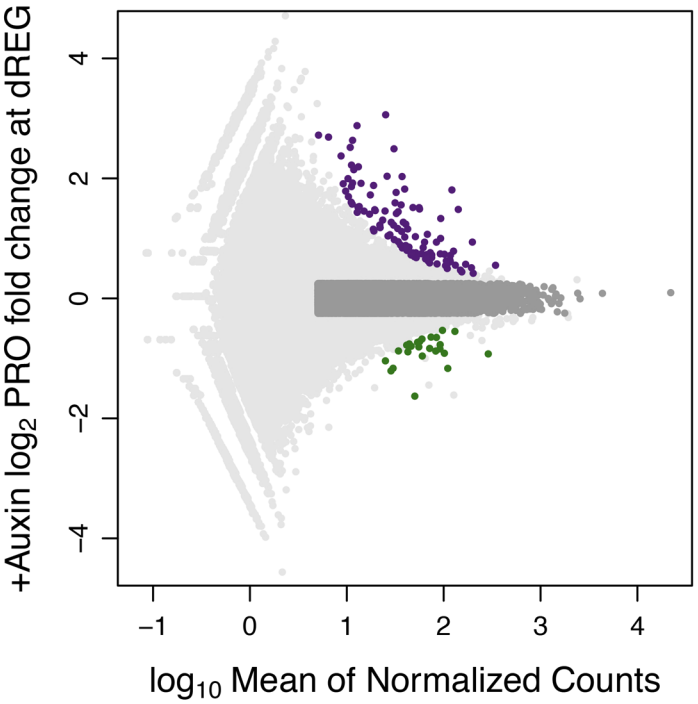

Figure 9: We observed modest changes in enhancer RNA expression after ZNF143 depletion.

4.1 Motif enrichment at dREG elements

Next we wanted to determine what motifs are enriched *de novo* in the increased and decreased regulatory elements. We hypothesize that if ZNF143 is a canonical activator, then the ZNF143 motif will be found *de novo* in the regulatory elements that decrease nascent RNA upon ZNF143 depletion. If ZNF143 is a repressor, then we expect the opposite. If ZNF143 is both an activator and repressor, then we would expect the motif to found in both classes. The first step is to retrieve a database of position weight matrices to query any *de novo* motifs against. We will use the HOMER database (Heinz *et al.*, 2010) and TOMTOM software (Bailey *et al.*, 2009).

Next we perform motif analysis using the sequence in an 200 base pair window centered on the dREG summit for the increased and decreased classes. Then we compare to the HOMER data base with TOMTOM.

```
#HOMER
wget wget https://raw.githubusercontent.com/mjg54/znf143_pro_seq_analysis/master/docs/HOMER_MEME_conversion.py
wget http://homer.ucsd.edu/homer/custom.motifs

python ~/ZNF143/HOMER_MEME_conversion.py -i custom.motifs -o custom.motifs

#file conversion
slopBed -i increased_df.HEK.auxin.arf.deseq.effects.dREG.lattice_0.1_FDR.bed \
    -g hg38.chrom.sizes -b 100 | fastaFromBed -fi hg38.fa -bed stdin \
    -fo increased_df.HEK.auxin.arf.deseq.effects.dREG.lattice_0.1_FDR.fasta

slopBed -i decreased_df.HEK.auxin.arf.deseq.effects.dREG.lattice_0.1_FDR.bed \
    -g hg38.chrom.sizes -b 100 | fastaFromBed -fi hg38.fa -bed stdin \
    -fo decreased_df.HEK.auxin.arf.deseq.effects.dREG.lattice_0.1_FDR.fasta

meme -o decreased_df.HEK.dREG.lattice_0.1_FDR.fasta.meme_chip_output -dna \
        -minw 5 -maxw 20 \
        decreased_df.HEK.auxin.arf.deseq.effects.dREG.lattice_0.1_FDR.fasta

meme -o increased_df.HEK.dREG.lattice_0.1_FDR.fasta.meme_chip_output -dna \
        -minw 5 -maxw 20 \
        increased_df.HEK.auxin.arf.deseq.effects.dREG.lattice_0.1_FDR.fasta

tomtom -png -o decreased_df.HEK.dREG.tomtom_output \
    decreased_df.HEK.dREG.lattice_0.1_FDR.fasta.meme_chip_output/meme.txt \
    custom.motifs_meme.txt

tomtom -o increased_df.HEK.dREG.tomtom_output \
    increased_df.HEK.dREG.lattice_0.1_FDR.fasta.meme_chip_output/meme.txt \
    custom.motifs_meme.txt
```

5 ChIP-seq analysis of ZNF143 binding.

Retrieve ZNF143 peaks from ENCODE (Consortium *et al.*, 2012).

```
encodeurl=http://hgdownload.cse.ucsc.edu/goldenpath/hg19/encodeDCC/wgEncodeSydhTfbs/

wget ${encodeurl}wgEncodeSydhTfbsK562Znf143IggrabPk.narrowPeak.gz
wget ${encodeurl}wgEncodeSydhTfbsGm12878Znf143166181apStdPk.narrowPeak.gz
wget http://hgdownload.soe.ucsc.edu/gbdb/hg19/liftOver/hg19ToHg38.over.chain.gz
gunzip hg19ToHg38.over.chain.gz

for peak in *narrowPeak.gz
do
    name=$(echo $peak | awk -F".narrowPeak" '{print $1}')
    unz=$(echo $peak | awk -F".gz" '{print $1}')
    echo $name
    gunzip $peak
    echo $unz
    liftOver $unz hg19ToHg38.over.chain $name.hg38.broadPeak $name.hg38.unmapped.txt -bedPlus=6
    awk '{OFS="\t";} {print $1,$2+$10,$2+$10+1}' $name.hg38.broadPeak > $name.summit.bed
done

cat *Znf143*.summit.bed > Znf143_K562_GM12878_peaks.bed
```

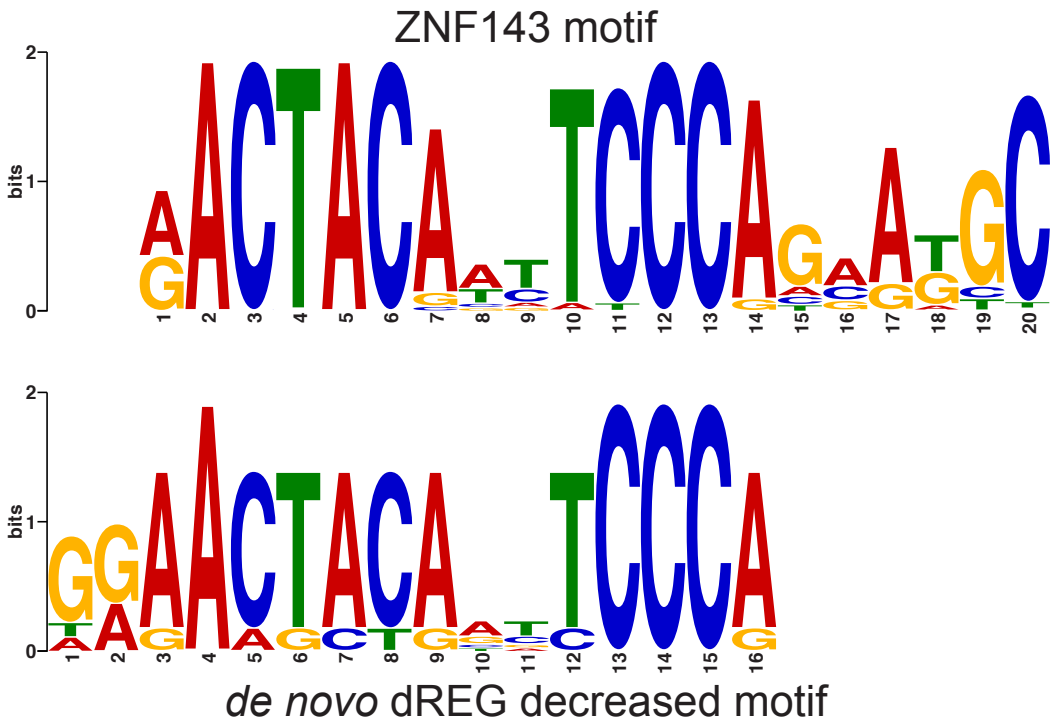

Figure 10: The most enriched motif in dREG-defined regulatory elements that decrease upon auxin depletion is the canonical ZNF143 motif.

```
sort -k1,1 -k2,2n Znf143_K562_GM12878_peaks.bed > Znf143_K562_GM12878_peaks.sorted.bed

mergeBed -i Znf143_K562_GM12878_peaks.sorted.bed > Znf143_K562_GM12878_peaks_merged.sorted.bed
```

5.1 ZNF143 and gene class distances

The following cumulative distribution function plots can imply activation or repression function of the transcription factor. The primary reference that first convinced me of an nuclear receptor acting exclusively in *cis* as an activator was Figure 2C of (Reddy *et al.*, 2009). Including the unchanged class is an important comparison because these genes are expressed at comparable levels to the activated and repressed class, see Figure 8.

```
#CDF plot rescue

df.chronic.effects$arfrescue = gsub('Auxin', 'ARF',
           df.chronic.effects$arfrescue)
df.chronic.effects[,17] = paste(df.chronic.effects$chronic, df.chronic.effects$arfrescue)

colnames(df.chronic.effects) = c("rownames", "baseMean.chronic", "log2FoldChange.chronic",
                                 "lfcSE.chronic", "stat.chronic", "pvalue.chronic",
                                "padj.chronic",  "arfrescue", "baseMean.rescue",
                                "log2FoldChange.rescue", "lfcSE.rescue", "stat.rescue",
                                "pvalue.rescue", "padj.rescue","auxrescue","rescue", "arfauxin")


df.all.chronic = cdf.deseq.df(df = df.chronic.effects[grep('Chronic Depletion Repressed ARF',
                                                           df.chronic.effects$arfauxin),],
           genes = gene.file, chip.peaks = 'Znf143_K562_GM12878_peaks_merged.sorted.bed',
           treat = 'Chronic Depletion Repressed ARF', tf.name = 'ZNF143' )

col.order = c("#FF0000","grey55", "#0000FF", "grey85" )
pdf(paste0(direc, "Figure_cdf_compare_chronic_Reg_classes_ZNF143_ARF.pdf"), width=6.2, height=3.83)

        col.lines = col.order
        ecdfplot(~log(abs(dis), base = 10), groups = status, data = df.all.chronic,
                 auto.key = list(lines=TRUE, points=FALSE),
```

```
                col = col.lines,
                aspect = 1,
                scales=list(relation="free",alternating=c(1,1,1,1)),
                ylab = 'Cumulative Distribution Function',
                xlab = expression('log'[10]~'ZNF143 Distance from TSS'),
                between=list(y=1.0),
                type = 'a',
                xlim = c(0,5.5),
                lwd=2,
                par.settings = list(superpose.line = list(col = col.lines, lwd=3),
                                    strip.background=list(col="grey85")),
                panel = function(...) {
                    panel.abline(v= 200, lty =2)
                    panel.ecdfplot(...)
                })
dev.off()
#nothing too informative here because the chronic depleted genes
#tend to be closer to CNF143 binding sites as a class



                                      #CDF plot

colnames(df.HEK.arf.rescue.deseq.effects.lattice) =
    c(colnames(df.HEK.arf.rescue.deseq.effects.lattice)[c(1:(length(colnames(df.HEK.arf.rescue.deseq.effects.lattice))-2))],
      'arfauxin', 'rescue')


df.HEK.arf.rescue.deseq.effects.lattice$arfauxin = gsub('Auxin', 'ARF',
            df.HEK.arf.rescue.deseq.effects.lattice$arfauxin)


df.all.arf = cdf.deseq.df(df = df.HEK.arf.rescue.deseq.effects.lattice,
            genes = gene.file, chip.peaks = 'Znf143_K562_GM12878_peaks_merged.sorted.bed',
            treat = 'ARF', tf.name = 'ZNF143_ARF')


pdf(paste0(direc, "Figure_cdf_compare_Reg_classes_ZNF143_ARF.pdf"), width=6.2, height=3.83)

        col.lines = c("#FF0000" , "grey60", "#0000FF","grey90")
        ecdfplot(~log(abs(dis), base = 10), groups = status, data = df.all.arf,
                auto.key = list(lines=TRUE, points=FALSE),
                col = col.lines,
                aspect = 1,
                scales=list(relation="free",alternating=c(1,1,1,1)),
                ylab = 'Cumulative Distribution Function',
                xlab = expression('log'[10]~'ZNF143 Distance from TSS'),
                between=list(y=1.0),
                type = 'a',
                xlim = c(0,5.5),
                lwd=2,
                par.settings = list(superpose.line = list(col = col.lines, lwd=3),
                                    strip.background=list(col="grey85")),
                panel = function(...) {
                    panel.abline(v= 200, lty =2)
                    panel.ecdfplot(...)
                })
dev.off()

df.HEK.arf.rescue.deseq.effects.lattice$arfauxin = gsub('ARF', 'Auxin',
                                                        df.HEK.arf.rescue.deseq.effects.lattice$arfauxin)



#pairwise comparisons
#
colnames(df.HEK.auxin.arf.deseq.effects.lattice) =
    c(colnames(df.HEK.auxin.arf.deseq.effects.lattice)[c(1:(length(colnames(df.HEK.auxin.arf.deseq.effects.lattice))-2))],
      'arfauxin', 'rescue')

df.all.auxin.arf = cdf.deseq.df(df = df.HEK.auxin.arf.deseq.effects.lattice,
                                    genes = gene.file, chip.peaks =
                                                        'Znf143_K562_GM12878_peaks_merged.sorted.bed',
                                    treat = 'Auxin ARF', tf.name = 'ZNF143_Auxin_ARF')


pdf(paste0(direc, "Figure_cdf_compare_Reg_classes_ZNF143_ARF_Auxin.pdf"), width=6.2, height=3.83)

        col.lines = c("#521e77" , "grey60", "#36771e","grey90")
        ecdfplot(~log(abs(dis), base = 10), groups = status, data = df.all.auxin.arf,
                auto.key = list(lines=TRUE, points=FALSE),
                col = col.lines,
                aspect = 1,
                scales=list(relation="free",alternating=c(1,1,1,1)),
                ylab = 'Cumulative Distribution Function',
                xlab = expression('log'[10]~'ZNF143 Distance from TSS'),
                between=list(y=1.0),
```

```
                type = 'a',
                xlim = c(0,5.5),
                lwd=2,
                par.settings = list(superpose.line = list(col = col.lines, lwd=3),
                                    strip.background=list(col="grey85")),
                panel = function(...) {
                    panel.abline(v= 200, lty =2)
                    panel.ecdfplot(...)
                })
dev.off()


colnames(df.HEK.auxin.deseq.effects.lattice) =
    c(colnames(df.HEK.auxin.deseq.effects.lattice)[c(1:(length(colnames(df.HEK.auxin.deseq.effects.lattice))-2))],
      'arfauxin', 'rescue')


df.all.auxin = cdf.deseq.df(df = df.HEK.auxin.deseq.effects.lattice,
                                    genes = gene.file, chip.peaks =
                                                        'Znf143_K562_GM12878_peaks_merged.sorted.bed',
                                    treat = 'Auxin', tf.name = 'ZNF143_Auxin')

pdf(paste0("Figure_cdf_compare_Reg_classes_ZNF143_Auxin.pdf"), width=6.2, height=3.83)

        col.lines = c("#D60909" , "grey50", "#0931D6","grey80")
        ecdfplot(~log(abs(dis), base = 10), groups = status, data = df.all.auxin,
                auto.key = list(lines=TRUE, points=FALSE),
                col = col.lines,
                aspect = 1,
                scales=list(relation="free",alternating=c(1,1,1,1)),
                ylab = 'Cumulative Distribution Function',
                xlab = expression('log'[10]~'ZNF143 Distance from TSS'),
                between=list(y=1.0),
                type = 'a',
                xlim = c(0,5.5),
                lwd=2,
                par.settings = list(superpose.line = list(col = col.lines, lwd=3),
                                    strip.background=list(col="grey85")),
                panel = function(...) {
                    panel.abline(v= 200, lty =2)
                    panel.ecdfplot(...)
                })
dev.off()



df.all.auxin.arf[, ncol(df.all.auxin.arf) + 1] = 'ARF Rescue'
df.all.auxin[, ncol(df.all.auxin) + 1] = 'Chronic Depletion'

colnames(df.all.auxin.arf) =
    c(colnames(df.all.auxin.arf)[1:ncol(df.all.auxin.arf) -1], 'rescue')


colnames(df.all.auxin) =
    c(colnames(df.all.auxin)[1:ncol(df.all.auxin) -1], 'rescue')

df.all.auxin.CDF = rbind(df.all.auxin.arf, df.all.auxin)

pdf("FIG_cdf_compare_Reg_classes_znf143.pdf", width=6.83, height=4.33)

col.lines = c("#521e77" , "grey60", "#36771e","grey90", "#D60909" , "grey50", "#0931D6","grey80")
ecdfplot(~log(abs(dis), base = 10) | rescue, groups = status, data = df.all.auxin.CDF,
         auto.key = list(lines=TRUE, points=FALSE),
         col = col.lines,
         aspect = 1,
         scales=list(relation="free",alternating=c(1,1,1,1)),
         ylab = expression('Cumulative Distribution Function'),
         xlab = expression('log'[10]~'ZNF143 Distance from TSS'),
         between=list(y=1.0),
         type = 'a',
         xlim = c(0,5.5),
         lwd=2,
         par.settings = list(superpose.line = list(col = col.lines, lwd=3),
                             strip.background=list(col="grey85")),
        panel = function(...) {
           panel.abline(v= 200, lty =2)
           panel.ecdfplot(...)
  })
dev.off()

save.image(file = paste0(direc, '181129_HEK_image.Rdata'))



ks.auxin.ActvUnc = ks.test(x = log(abs(df.all.auxin$dis)[df.all.auxin$status == 'Auxin Activated']),
        y = log(abs(df.all.auxin$dis)[df.all.auxin$status == 'Auxin Unchanged']))$p.value
ks.auxin.RepvUnc = ks.test(x = log(abs(df.all.auxin$dis)[df.all.auxin$status == 'Auxin Repressed']),
```

```
        y = log(abs(df.all.auxin$dis)[df.all.auxin$status == 'Auxin Unchanged']))$p.value
ks.auxin.arf.ActvUnc = ks.test(x = log(abs(df.all.auxin.arf$dis)[df.all.auxin.arf$status ==
                                                                 'Auxin ARF Activated']),
        y = log(abs(df.all.auxin.arf$dis)[df.all.auxin.arf$status == 'Auxin ARF Unchanged']))$p.value

ks.auxin.arf.RepvUnc = ks.test(x = log(abs(df.all.auxin.arf$dis)[df.all.auxin.arf$status ==
                                                                 'Auxin ARF Repressed']),
        y = log(abs(df.all.auxin.arf$dis)[df.all.auxin.arf$status == 'Auxin ARF Unchanged']))$p.value
ks.auxin.arf.RepvUnc = formatC(ks.auxin.arf.RepvUnc, format = "e", digits = 3)

save(ks.auxin.ActvUnc, ks.auxin.RepvUnc, ks.auxin.arf.ActvUnc,
     ks.auxin.arf.RepvUnc, file = paste0(direc, 'ks.auxin.Rdata'))
```

We find that ZNF143 binding, as measured by ChIP-seq, is enriched proximal to the repressed gene class in the ARF rescue background (Kolmogorov–Smirnov two-sided p-value = $1.110e-16$) and not the activated gene class (Kolmogorov–Smirnov two-sided p-value = 0.0470157). In the chronic ZNF143 depletion background auxin respressed genes are, on average, not significantly closer to ZNF143 binding sites (Kolmogorov–Smirnov two-sided p-value = 0.0219449) and the activated gene class tend to be further away from ZNF143 binding sites (Kolmogorov–Smirnov two-sided p-value = 0.0017018).

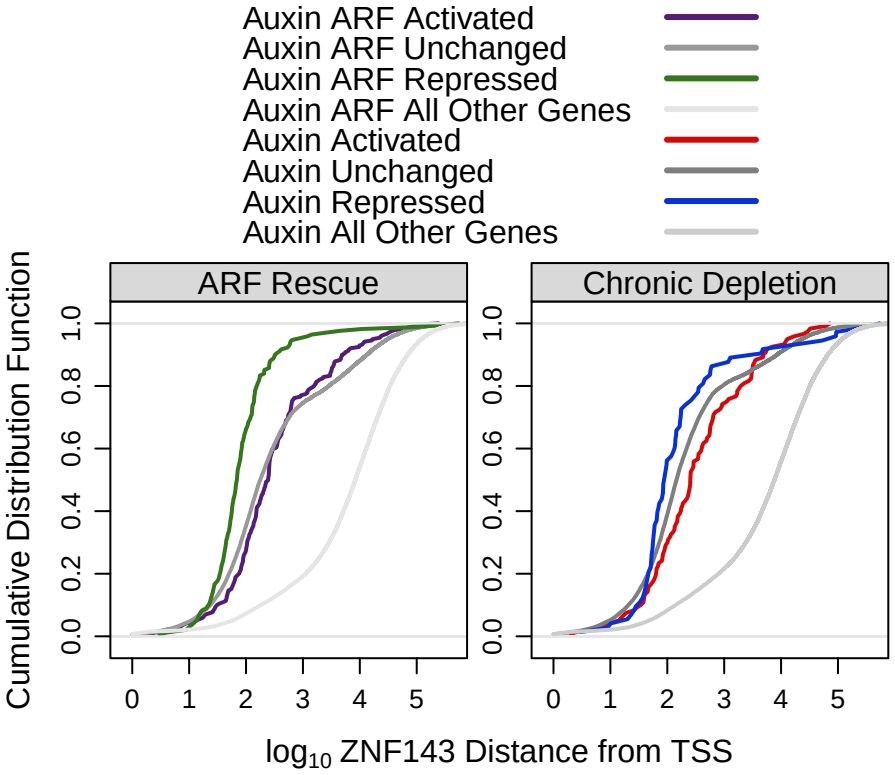

Figure 11: ZNF143 binding proximity to gene classes in both the chronic depletion and rescue backgrounds.

Combine replicates to load into a browser and use `bigWig` or `bedGraph` files for plotting composite profiles.

```
for i in HEK_*_rep*PE1_pro_*.bam
do
    name=$(echo $i | awk -F"/" '{print $NF}' | awk -F"_PE1_pro_" '{print $1}')
    strand=$(echo $i | awk -F"pro_" '{print $NF}' | awk -F".bam" '{print $1}')
    invscale1=$(grep ${name} norm.bedGraph.sizeFactor.txt | awk -F" " '{print $2}')
    invscale=$(echo $invscale1 |bc)
    scaletrue=$(bc <<< "scale=4 ; 1.0 / $invscale")
    echo $name
    echo $strand
    echo $scaletrue
    echo file_to_scale
    echo ${name}_PE1_${strand}_body_0-mer.bigWig
    bigWigToBedGraph ${name}_PE1_${strand}_body_0-mer.bigWig ${name}_${strand}_body_0-mer.bg
    echo normalizing
    python ~/ZNF143/normalize_bedGraph.py -i ${name}_${strand}_body_0-mer.bg \
       -s $scaletrue -o ${name}_${strand}_scaled.bg
    bedGraphToBigWig ${name}_${strand}_scaled.bg hg38.chrom.sizes ${name}_pro_${strand}_scaled.bigWig
    rm ${name}_${strand}_scaled.bg
    rm ${name}_${strand}_body_0-mer.bg
done
```

Next merge bigWigs of the same condition

```
for i in *rep1*_scaled.bigWig
do
    name=$(echo $i | awk -F"/" '{print $NF}' | awk -F"_rep1" '{print $1}')
    strand=$(echo $i | awk -F"pro_" '{print $NF}' | awk -F"_scaled.bigWig" '{print $1}')
    echo $name
    echo $strand
    files=$(ls ${name}_rep*${strand}_scaled.bigWig)
    echo $files
    touch temp.txt
    if [ "$strand" == "plus" ]
    then
        echo "track type=bedGraph name=${name}_plus color=255,0,0 alwaysZero=on visibility=full" >> temp.txt
    fi
    if [ "$strand" == "minus" ]
    then
        echo "track type=bedGraph name=${name}_minus color=0,0,255 alwaysZero=on visibility=full" >> temp.txt
    fi
    count=$(ls ${name}_rep*${strand}_scaled.bigWig | wc -l | bc)
    scaleall=$(bc <<< "scale=4 ; 1.0 / $count")
    echo $count
    echo $scaleall
    bigWigMerge $files tmp.bg
    python ~/ZNF143/normalize_bedGraph.py -i tmp.bg \
       -s $scaleall -o ${name}_${strand}_scaled.bg
    LC_COLLATE=C sort -k1,1 -k2,2n ${name}_${strand}_scaled.bg > ${name}_${strand}_scaled_sorted.bg
    cat temp.txt ${name}_${strand}_scaled_sorted.bg > ${name}_${strand}_scaled.bedGraph
    rm tmp.bg
    rm temp.txt
    rm ${name}_${strand}_scaled.bg
    rm ${name}_${strand}_scaled_sorted.bg
    head ${name}_${strand}_scaled.bedGraph
    bedGraphToBigWig ${name}_${strand}_scaled.bedGraph hg38.chrom.sizes \
        ${name}_${strand}_combined_normalized.bigWig
    gzip ${name}_${strand}_scaled.bedGraph
done
```

6 Deeper analyses in R

6.1 Testing whether the rescue does have increased sensitivity and specificity

```
df.auxin.effects = merge(df.HEK.auxin.deseq.effects.lattice, df.HEK.auxin.arf.deseq.effects.lattice,
                         by="row.names")

rownames(df.auxin.effects) = df.auxin.effects[,1]
colnames(df.auxin.effects) = c("rownames", "baseMean.auxin", "log2FoldChange.auxin",
                               "lfcSE.auxin", "stat.auxin", "pvalue.auxin",
                             "padj.auxin",  "auxin", "auxinalone", "baseMean.arfauxin",
                             "log2FoldChange.arfauxin", "lfcSE.arfauxin", "stat.arfauxin",
                             "pvalue.arfauxin", "padj.arfauxin","arfauxin","arfrescue")

nrow(df.auxin.effects[df.auxin.effects$arfauxin == 'Auxin ARF Repressed' &
                      df.auxin.effects$auxin == 'Auxin Repressed',])

nrow(df.auxin.effects[df.auxin.effects$arfauxin == 'Auxin ARF Repressed',])

nrow(df.auxin.effects[df.auxin.effects$auxin == 'Auxin Repressed',])

df.auxin.effects[!(df.auxin.effects$arfauxin == 'Auxin ARF Repressed') &
                 df.auxin.effects$auxin == 'Auxin Repressed',]

mag.response = (2^df.auxin.effects$log2FoldChange.arfauxin[df.auxin.effects$auxin == 'Auxin Repressed']) /
    (2^df.auxin.effects$log2FoldChange.auxin[df.auxin.effects$auxin == 'Auxin Repressed'])
mag.response.2 = df.auxin.effects$arfauxin[df.auxin.effects$auxin == 'Auxin Repressed']


pdf(paste0(direc, 'Ratio_of_auxin_changes_in_Auxin_Repressed_Class_bwplot.pdf'), width=6.4, height=6.0)

col.order = c('grey90', "grey60","#36771e") #purple: #521e77
trellis.par.set(box.umbrella = list(lty = 1, col=col.order, lwd=3),
                box.rectangle = list(col = col.order, lwd=3), #"#D60909" , "grey50", "#0931D6","grey80")
                plot.symbol = list(col=c("black"), lwd=1.0, pch ='.'))

bwplot(log(as.numeric(as.character(mag.response)), base = 2) ~ mag.response.2,
       horizontal =FALSE,
       between=list(y=1.0, x = 1.0),
       main = 'All Auxin Repressed Genes upon Chronic ZNF143 Depletion',
       scales=list(relation="free",rot = 45, alternating=c(1,1,1,1),
                   x = list(col = col.order)),
       ylab = expression('log'[2]*' (ARF Rescue Auxin Response / Auxin Response)'),
       xlab = expression('ARF Rescue Auxin Response Category'),
        pch = '|', col= 'black',
       aspect = 1.5,
       do.out = FALSE,
       par.settings=list(par.xlab.text=list(cex=1.0,font=1),
           par.ylab.text=list(cex=1.0,font=1),
           par.main.text=list(cex=1.0, font=1)),
       strip = function(..., which.panel, bg) {
           bg.col = c("#ce228e" , "#2290cf","grey90","grey60")
           strip.default(..., which.panel = which.panel,
                         bg = rep(bg.col, length = which.panel)[which.panel])
       },
       panel = function(..., box.ratio, col, pch) {
           panel.abline(h = 0, col = 'grey65')
           panel.violin(..., col = 'white',
                        varwidth = FALSE, box.ratio = box.ratio)
           panel.stripplot(..., col='grey20', do.out=FALSE, jitter.data=TRUE,pch = 20)
           panel.bwplot(..., col = col.order)

       })
dev.off()


#look at the raw auxin reponse in ARF Rescue Auxin Repressed genes
mag.response.3 = df.auxin.effects$log2FoldChange.auxin[df.auxin.effects$arfauxin == 'Auxin ARF Repressed']
mag.response.4 = df.auxin.effects$auxin[df.auxin.effects$arfauxin == 'Auxin ARF Repressed']

pdf(paste0(direc, 'Raw_auxin_changes_in_ARF_Rescue_Auxin_repressed_class_bwplot.pdf'),
    useDingbats = FALSE, width=6.4, height=6.0)
col.order = c('grey80', "grey50","#0931D6")
trellis.par.set(box.umbrella = list(lty = 1, col=col.order, lwd=3),
                box.rectangle = list(col = col.order, lwd=3), #"#D60909" , "grey50", "#0931D6","grey80")
                plot.symbol = list(col=c("black"), lwd=1.0, pch ='.'))

bwplot(mag.response.3 ~ mag.response.4,
       horizontal =FALSE,
       between=list(y=1.0, x = 1.0),
       scales=list(relation="free",rot = 45, alternating=c(1,1,1,1), x = list(col = col.order)),
       main = 'All Auxin Repressed Genes upon ARF Rescue',
       ylab = expression('log'[2]*' Chronic ZNF143 Depletion Auxin Response'),
```

```
       xlab = expression('Chronic ZNF143 Depletion Auxin Response Category'),
       pch = '|', col= 'black',
       aspect = 1.5,
       do.out = FALSE,
       par.settings=list(par.xlab.text=list(cex=1.0,font=1),
           par.ylab.text=list(cex=1.0,font=1),
           par.main.text=list(cex=1.0, font=1)),
       strip = function(..., which.panel, bg) {
           #bg.col = c("#ce228e" , "#2290cf","grey90","grey60")
           strip.default(..., which.panel = which.panel, bg = rep(bg.col, length = which.panel)[which.panel])
       },
       panel = function(..., box.ratio, col, pch) {
           panel.abline(h = 0, col = 'grey65')
           panel.violin(..., col=c('white'),
                        varwidth = FALSE, box.ratio = box.ratio)
           panel.stripplot(..., col="grey20",
                           do.out=FALSE, jitter.data=TRUE,pch = 20)
           panel.bwplot(..., col = col.order)

       })
dev.off()



mag.response.5 =
    (2^df.auxin.effects$log2FoldChange.arfauxin[df.auxin.effects$arfauxin == 'Auxin ARF Repressed']) /
    (2^df.auxin.effects$log2FoldChange.auxin[df.auxin.effects$arfauxin == 'Auxin ARF Repressed'])
mag.response.6 = df.auxin.effects$auxin[df.auxin.effects$arfauxin == 'Auxin ARF Repressed']


pdf(paste0(direc, 'Ratio_of_auxin_changes_in_Chronic_Depletion_Auxin_Repressed_Class_bwplot.pdf'),
    width=6.4, height=6.0)
col.order = c('grey80', "grey50","#0931D6")
trellis.par.set(box.umbrella = list(lty = 1, col=col.order, lwd=3),
                box.rectangle = list(col = col.order, lwd=3), #"#D60909" , "grey50", "#0931D6","grey80")
                plot.symbol = list(col=c("black"), lwd=1.0, pch ='.'))

bwplot(log(as.numeric(as.character(mag.response.5)), base = 2) ~ mag.response.6,
       horizontal =FALSE,
       between=list(y=1.0, x = 1.0),
       main = 'All Auxin Repressed upon ARF Rescue',
       scales=list(relation="free",rot = 45, alternating=c(1,1,1,1), x = list(col = col.order)),
       ylab = expression('log'[2]*' (ARF Rescue Auxin Response / Auxin Response)'),
       xlab = expression('Chronic ZNF143 Depletion Auxin Response Category'),
       pch = '|',
       col= 'black',
       aspect = 1.5,
       do.out = FALSE,
       par.settings=list(par.xlab.text=list(cex=1.0,font=1),
           par.ylab.text=list(cex=1.0,font=1),
           par.main.text=list(cex=1.0, font=1)),
       strip = function(..., which.panel, bg) {
           bg.col = c("#ce228e" , "#2290cf","grey90","grey60")
           strip.default(..., which.panel = which.panel, bg = rep(bg.col, length = which.panel)[which.panel])
       },
       panel = function(..., box.ratio, col, pch) {
           panel.abline(h = 0, col = 'grey65')
           panel.violin(..., col=c('white'),
                        varwidth = FALSE, box.ratio = box.ratio)
           panel.stripplot(..., col="grey20",
                           do.out=FALSE, jitter.data=TRUE,pch = 20)
           panel.bwplot(..., col = col.order)

       })
dev.off()

aux.arf.repressed.number =
    length(df.auxin.effects$arfauxin[df.auxin.effects$arfauxin == 'Auxin ARF Repressed'])
length(mag.response[log(as.numeric(as.character(mag.response)), base = 2) < 0])
length(mag.response[log(as.numeric(as.character(mag.response)), base = 2) > 0])

x = ((2^df.auxin.effects$log2FoldChange.arfauxin[df.auxin.effects$auxin == 'Auxin Repressed' &
                                                   df.auxin.effects$arfauxin == 'Auxin ARF Repressed' ]) /
    (2^df.auxin.effects$log2FoldChange.auxin[df.auxin.effects$auxin == 'Auxin Repressed'&
                                               df.auxin.effects$arfauxin == 'Auxin ARF Repressed' ]))
length(x[x<1])
length(x[x>1])


aux.chronic.repressed.number = length(df.auxin.effects$auxin[df.auxin.effects$auxin == 'Auxin Repressed'])
dually.repressed.class = length(x[x<1]) + length(x[x>1])#57
more.rep.in.chronic = length(x[x<1])
aux.raw.negative.arf.rep = length(mag.response.3[mag.response.3 < 0])#167

length(mag.response.3[mag.response.3 > 0])
```

```
greater.mag.in.rescue = length(mag.response.5[log(as.numeric(as.character(mag.response.5)), base = 2) < 0])
length(mag.response.5[log(as.numeric(as.character(mag.response.5)), base = 2) > 0])

save(aux.arf.repressed.number, aux.raw.negative.arf.rep,
     greater.mag.in.rescue, dually.repressed.class,
     aux.chronic.repressed.number, more.rep.in.chronic,
     file= paste0(direc, 'sen_spec_comparison_numbers.Rdata'))
```

Of the 168 genes that are within the Repressed class in the Rescue background, 167 have a net negative change in gene expression upon auxin treatment in the chronic depletion background Figure 12. 146 of the 168 genes have a greater magnitude of response in rescue compared to chronic depletion Figure 13. These data suggest that the ARF rescue is more sensitive to detect changes in ZNF143-dependent transcription and the fact that the magnitude of these responses is enhances indicates that it is not simply resulting from differential variation between experimental conditions affecting the number of genes falling below FDR thresholds.

57 of the 73 Auxin Repressed genes class are categorized as Repressed in the rescue as well (Figure 14). 40 of the 57 are more repressed to a greater magnitude in the rescue (Figure 14). This analysis suggests that the ARF rescue is more responsive to ZNF143 depletion. This is consistent with Figure 11, in that the repressed genes in the chronic depletion are not significantly closer, on average, to ZNF143 binding sites.

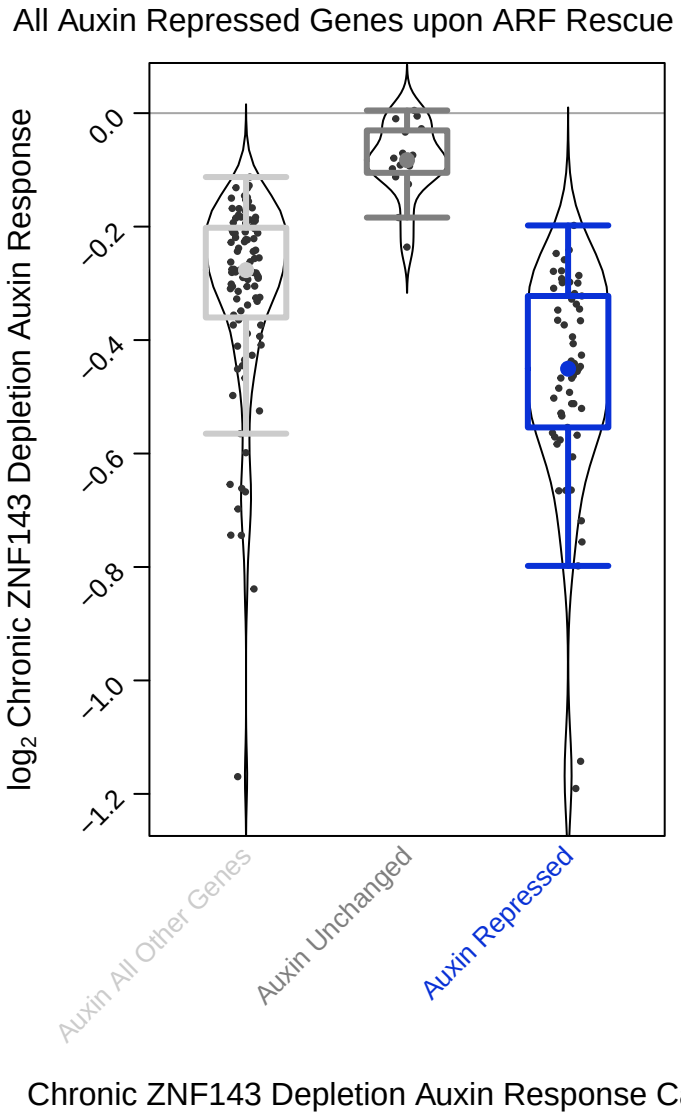

Figure 12: Despite falling in Unchanged and All Other Genes classes, the net reponse upon auxin treatment in the chronic depletion background is consistent in direction with auxin response in the rescue.

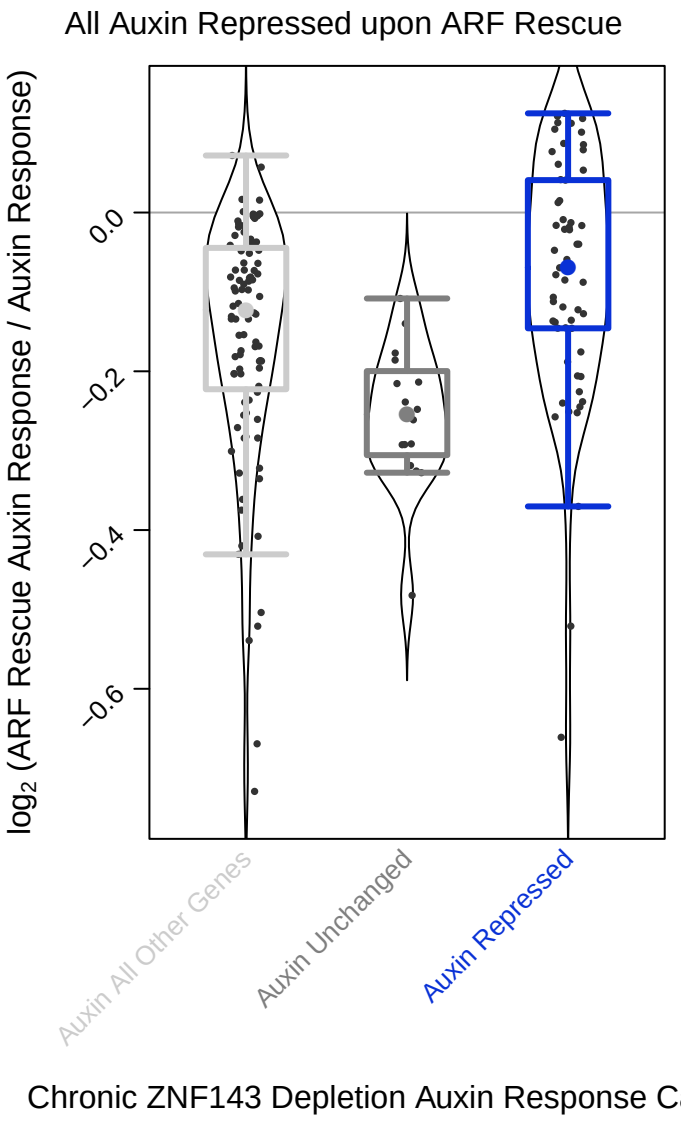

Figure 13: Genes have a greater magnitude of response in rescue compared to chronic depletion.

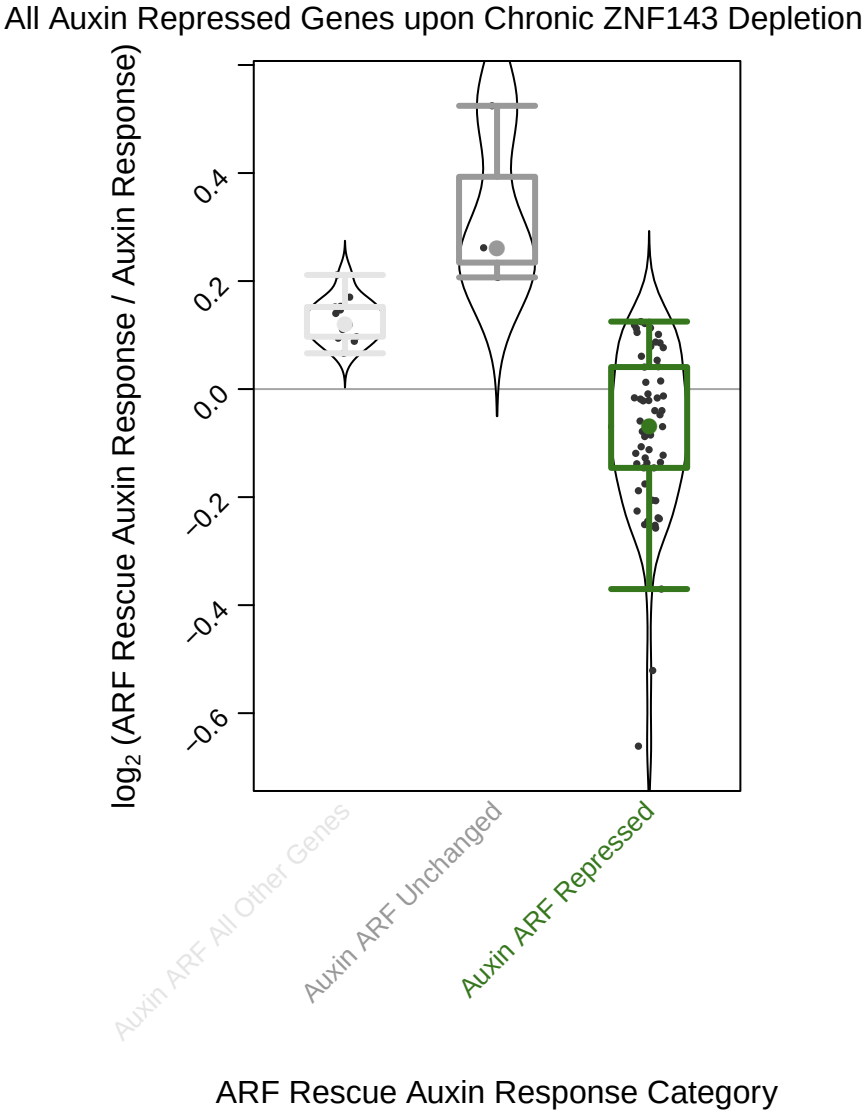

Figure 14: Repressed genes in the chronic depletion background are not consistently auxin repressed in the rescue and the rescue response in not consistently a greater magnitude when the gene falls in the repressed class in both backgrounds.

6.2 Composite and heatmap profiles

Plot composites

```
auxin.categories =
    list(df.HEK.auxin.arf.deseq.effects.lattice[df.HEK.auxin.arf.deseq.effects.lattice$arfauxin ==
                                    'Auxin ARF Unchanged',],

         df.HEK.auxin.arf.deseq.effects.lattice[df.HEK.auxin.arf.deseq.effects.lattice$arfauxin ==
                                    'Auxin ARF Activated',],
         df.HEK.auxin.arf.deseq.effects.lattice[df.HEK.auxin.arf.deseq.effects.lattice$arfauxin ==
                                    'Auxin ARF Repressed',],
         df.HEK.auxin.arf.deseq.effects.lattice[df.HEK.auxin.arf.deseq.effects.lattice$arfauxin ==
                                    'Auxin ARF All Other Genes',])

auxin.names = c('Auxin ARF Unchanged', 'Auxin ARF Activated', 'Auxin ARF Repressed',
                'Auxin ARF All Other Genes')


gene.file.tss = pro.composites(gene.file)

lat.df=data.frame(matrix(ncol = 7, nrow = 0))
colnames(lat.df) = c('est', 'x', 'cond', 'upper', 'lower', 'grp', 'auxin')

for (i in 1:4) {
    print(dim(auxin.categories[[i]])[1])
    out.x = streamplot(auxin.categories[[i]], direc, reg = 2000, stp = 10, genelist = gene.file.tss,
                       class =auxin.names[i], file.prefix = 'HEK_TIR1_ZNF143AID_ARF_')
    out.x[[1]]$auxin = auxin.names[i]
    lat.df = rbind(lat.df, out.x[[1]])
}



save(lat.df, file = paste0(direc, 'lat.df.hek.2000.10.Rdata'))
load(paste0(direc,'lat.df.hek.2000.10.Rdata'))


reg = 2000



composites.func.panels.pro.lattice(lat.df[lat.df$grp == 'Minus',],'RNA Polymerase',
                                   summit = 'TSS', num.m = -1800,
                                   num.p = 200, y.low = 0.005, y.high = 0.1, class = 'All_plots_Minus',
                                   col.lines = c('#616161CC','#FF0000CC'),
                                   fill.poly = c('#61616100','#FF000000'))
composites.func.panels.pro.lattice(lat.df[lat.df$grp == 'Plus',],'RNA Polymerase',
                                   summit = 'TSS', num.m = -200,
                                   num.p = 1800, y.low = 0.005, y.high = 0.5, class = 'All_plots_Plus',
                                   col.lines = c('#616161CC', '#FF0000CC'),
                                   fill.poly = c('#61616100','#FF000000'))


composites.func.panels.pro.lattice(lat.df[lat.df$grp == 'Plus',],'RNA Polymerase',
                                   summit = 'TSS', num.m = -100,
                                   num.p = 300, y.low = 0.005, y.high = 0.5,
                                   class = 'All_plots_Plus_zoom',
                                   col.lines = c( '#616161CC', '#FF0000CC'),
                                   fill.poly = c('#61616100','#FF000000'))


composites.func.Repressed(lat.df[lat.df$grp == 'Plus' & lat.df$auxin == 'Auxin ARF Repressed',],
                          'RNA Polymerase', summit = 'TSS', num.m = -200, num.p = 1800,
                          y.low = 0.03, y.high = 0.5, class = 'Repressed_Plus',
                          col.lines = c('#616161CC','#FF0000CC'))

composites.func.Repressed(lat.df[lat.df$grp == 'Plus' & lat.df$auxin == 'Auxin ARF Repressed',],
                          'RNA Polymerase', summit = 'TSS_', num.m = -100, num.p = 280,
                          y.low = 0.03, y.high = 0.5, class = 'Repressed_Plus',
                          col.lines = c('#616161CC','#FF0000CC'))
```

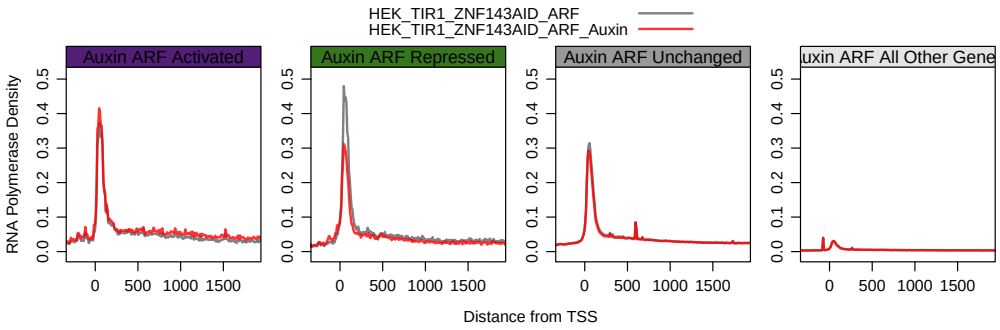

Figure 15: RNA Polymerase pause density is reduced in the repressed gene class with only modest differences in the other classes.

6.3 Pausing analysis

```
#loaded.bw.plus = load.bigWig(paste0(dir,
#                                    'combined_bigWig/HEK_plus_combined_no_scale.bigWig'))
#loaded.bw.minus = load.bigWig(paste0(dir,
#                                    'combined_bigWig/HEK_minus_combined_no_scale.bigWig'))

pause.read.counts = raw.pause.interval.slide(gene.file, direc, loaded.bw.plus,
                                             loaded.bw.minus, file.prefix = 'HEK_TIR1_ZNF143AID_ARF')

colnames(pause.read.counts) = sapply(strsplit(colnames(pause.read.counts), '_PE1'), '[', 1)


sample.conditions = factor(c("untreated","untreated", "untreated","untreated",
                             "treated","treated", "treated","treated"),
                           levels=c("untreated","treated"))

dds.prc = DESeqDataSetFromMatrix(pause.read.counts[,c(
    'HEK_TIR1_ZNF143AID_ARF_rep1',
    'HEK_TIR1_ZNF143AID_ARF_rep2',
    'HEK_TIR1_ZNF143AID_ARF_rep3',
    'HEK_TIR1_ZNF143AID_ARF_rep4',
    'HEK_TIR1_ZNF143AID_ARF_Auxin_rep1',
    'HEK_TIR1_ZNF143AID_ARF_Auxin_rep2',
    'HEK_TIR1_ZNF143AID_ARF_Auxin_rep3',
    'HEK_TIR1_ZNF143AID_ARF_Auxin_rep4')],
    DataFrame(sample.conditions), ~ sample.conditions)

pause.size.factors = estimateSizeFactorsForMatrix(df.HEK[,c(
    'HEK_TIR1_ZNF143AID_ARF_rep1',
    'HEK_TIR1_ZNF143AID_ARF_rep2',
    'HEK_TIR1_ZNF143AID_ARF_rep3',
    'HEK_TIR1_ZNF143AID_ARF_rep4',
    'HEK_TIR1_ZNF143AID_ARF_Auxin_rep1',
    'HEK_TIR1_ZNF143AID_ARF_Auxin_rep2',
    'HEK_TIR1_ZNF143AID_ARF_Auxin_rep3',
    'HEK_TIR1_ZNF143AID_ARF_Auxin_rep4')])

sizeFactors(dds.prc) = pause.size.factors
dds.prc = estimateDispersions(dds.prc)
dds.prc = nbinomWaldTest(dds.prc)
res.dds.prc = results(dds.prc)

plot.ma.lattice(res.dds.prc, filename = 'pause.test', p = 0.05,log2fold = 0.0)

df.auxin.effects.pause = merge(df.auxin.effects, res.dds.prc,
                               by="row.names")

fdr = 0.01

df.pause.ma =  as.data.frame(df.auxin.effects.pause)
df.pause.ma[,ncol(df.pause.ma) +1] = as.character(df.pause.ma$arfauxin)
df.pause.ma[df.pause.ma$padj < fdr & !is.na(df.pause.ma$padj), ncol(df.pause.ma)] = 'significant'


pdf(paste("MA_plot_Auxin_pause_sig_response_in_classes_groups_fdr_", fdr, ".pdf", sep=''),
        useDingbats = FALSE, width=10.83, height=5.83);
print(xyplot(df.pause.ma$log2FoldChange ~ log(df.pause.ma$baseMean, base=10) |
                df.pause.ma$arfauxin,
             groups=df.pause.ma[,ncol(df.pause.ma)],
             col=c( "#521e77", 'grey90', "#36771e", "grey60",'dark red'),
             main='Auxin Responsive Classes of Genes',
             scales="free",
             aspect=1,
             index.cond = list(c(3,4,2,1)),
             ylim=c(-5, 5),
             xlim=c(-1,3),
             pch=20,
             cex=0.5,
             ylab=expression("log"[2]~"Pause fold difference upon Auxin treatment"),
             xlab=expression("log"[10]~"Mean of Normalized Counts"),strip = function(..., which.panel, bg) {
                 bg.col = c("grey90", "grey60", "#521e77", "#36771e")
                 strip.default(..., which.panel = which.panel,
                               bg = rep(bg.col, length = which.panel)[which.panel])
             },
            par.settings=list(par.xlab.text=list(cex=1.1,font=2),
                              par.ylab.text=list(cex=1.1,font=2), strip.background=list(col="grey85")),
             panel = function(x, y, ...){
                   panel.xyplot(x, y, ...)
                   panel.abline(h = 0, col = 'grey65', lty =2)
             }
            ))
dev.off()
```

```
body.read.counts = raw.body.interval(gene.file, direc, file.prefix = 'HEK_TIR1_ZNF143AID_ARF')

colnames(body.read.counts) = sapply(strsplit(colnames(body.read.counts), '_PE1'), '[', 1)


sample.conditions = factor(c("untreated","untreated", "untreated","untreated",
                             "treated","treated","treated","treated"),
                           levels=c("untreated","treated"))


dds.brc = DESeqDataSetFromMatrix(body.read.counts[,c(
    'HEK_TIR1_ZNF143AID_ARF_rep1',
    'HEK_TIR1_ZNF143AID_ARF_rep2',
    'HEK_TIR1_ZNF143AID_ARF_rep3',
    'HEK_TIR1_ZNF143AID_ARF_rep4',
    'HEK_TIR1_ZNF143AID_ARF_Auxin_rep1',
    'HEK_TIR1_ZNF143AID_ARF_Auxin_rep2',
    'HEK_TIR1_ZNF143AID_ARF_Auxin_rep3',
    'HEK_TIR1_ZNF143AID_ARF_Auxin_rep4')], DataFrame(sample.conditions), ~ sample.conditions)


body.size.factors = estimateSizeFactorsForMatrix(df.HEK[,c(
    'HEK_TIR1_ZNF143AID_ARF_rep1',
    'HEK_TIR1_ZNF143AID_ARF_rep2',
    'HEK_TIR1_ZNF143AID_ARF_rep3',
    'HEK_TIR1_ZNF143AID_ARF_rep4',
    'HEK_TIR1_ZNF143AID_ARF_Auxin_rep1',
    'HEK_TIR1_ZNF143AID_ARF_Auxin_rep2',
    'HEK_TIR1_ZNF143AID_ARF_Auxin_rep3',
    'HEK_TIR1_ZNF143AID_ARF_Auxin_rep4')])

sizeFactors(dds.brc) = body.size.factors
dds.brc = estimateDispersions(dds.brc)
dds.brc = nbinomWaldTest(dds.brc)
res.dds.brc = results(dds.brc)

plot.ma.lattice(res.dds.brc, filename = 'body.test')


df.auxin.effects.body = merge(df.auxin.effects, res.dds.brc,
                              by="row.names")


pdf("MA_plot_Auxin_body_response_in_classes_groups.pdf", useDingbats = FALSE,width=10.83, height=5.83);
print(xyplot(df.auxin.effects.body$log2FoldChange ~ log(df.auxin.effects.body$baseMean, base=10) |
                df.auxin.effects.body$arfauxin,
             groups=df.auxin.effects.body$arfauxin,
             col=c("grey90", "grey60", "#521e77", "#36771e"),
             main='Auxin Responsive Classes of Genes',
             scales="free",
             aspect=1,
             index.cond = list(c(3,4,2,1)),
             ylim=c(-5, 5),
             xlim=c(-1,4),
             pch=20,
             cex=0.5,
             ylab=expression("log"[2]~"Pause fold difference upon Auxin treatment"),
             xlab=expression("log"[10]~"Mean of Normalized Counts"),strip = function(..., which.panel, bg) {
                 bg.col = c("grey90", "grey60", "#521e77", "#36771e")
                 strip.default(..., which.panel = which.panel,
                               bg = rep(bg.col, length = which.panel)[which.panel])
             },
            par.settings=list(par.xlab.text=list(cex=1.1,font=2),
                              par.ylab.text=list(cex=1.1,font=2), strip.background=list(col="grey85")),
             panel = function(x, y, ...){
                   panel.xyplot(x, y, ...)
                   panel.abline(h = 0, col = 'grey65', lty =2)
             }
            ))
dev.off()


fdr = 0.0001

df.body.ma =  as.data.frame(df.auxin.effects.body)
df.body.ma[,ncol(df.body.ma) +1] = as.character(df.body.ma$arfauxin)
df.body.ma[df.body.ma$padj < fdr & !is.na(df.body.ma$padj), ncol(df.body.ma)] = 'significant'

pdf(paste("MA_plot_Auxin_body_sig_response_in_classes_groups_fdr_", fdr, ".pdf", sep=''),
      useDingbats = FALSE, width=10.83, height=5.83);
print(xyplot(df.body.ma$log2FoldChange ~ log(df.body.ma$baseMean, base=10) |
                df.body.ma$arfauxin,
             groups=df.body.ma[,ncol(df.body.ma)],
             col=c( "#521e77", 'grey90', "#36771e", "grey60",'dark red'),
             main='Auxin Responsive Classes of Genes',
             scales="free",
```

```
             aspect=1,
             index.cond = list(c(3,4,2,1)),
             ylim=c(-5, 5),
             xlim=c(-1,4),
             pch=20,
             cex=0.5,
             ylab=expression("log"[2]~"Body fold difference upon Auxin treatment"),
             xlab=expression("log"[10]~"Mean of Normalized Counts"),strip = function(..., which.panel, bg) {
                 bg.col = c("grey90", "grey60", "#521e77", "#36771e")
                 strip.default(..., which.panel = which.panel,
                               bg = rep(bg.col, length = which.panel)[which.panel])
             },
            par.settings=list(par.xlab.text=list(cex=1.1,font=2),
                              par.ylab.text=list(cex=1.1,font=2), strip.background=list(col="grey85")),
             panel = function(x, y, ...){
                   panel.xyplot(x, y, ...)
                   panel.abline(h = 0, col = 'grey65', lty =2)
             }
            ))
dev.off()


                                        #one panel for main
x = df.pause.ma[df.pause.ma$arfauxin == 'Auxin ARF Repressed',]
y =  df.body.ma[df.body.ma$arfauxin == 'Auxin ARF Repressed',]

x[,ncol(x) + 1 ] = 'Pause Region'
y[,ncol(y) + 1] = 'Gene Body'

df.act.genes.ma = rbind(x,y)

pdf(paste(direc, "MA_plot_Auxin_body_pause_Repressed_fdr_", fdr, ".pdf", sep=''),
    useDingbats = FALSE, width=4.83, height=3.83);
print(xyplot(df.act.genes.ma$log2FoldChange ~ log(df.act.genes.ma$baseMean, base=10) |
                 df.act.genes.ma[,ncol(df.act.genes.ma)],
             groups=df.act.genes.ma[,ncol(df.act.genes.ma) -1 ],
             col=c("#36771e", 'dark red'),
             main='ARF Rescue Auxin Repressed Gene Class',
             scales="free",
             aspect=1,
             ylim=c(-3, 3),
             xlim=c(-0.5, 4.1),
             index.cond = list(c(2,1)),
             pch=20,
             cex=0.5,
             ylab=expression("log"[2]~"PRO change upon Auxin"),
             xlab=expression("log"[10]~"Mean of Normalized Counts"),
            par.settings=list(par.xlab.text=list(cex=1.1,font=2),
                              par.ylab.text=list(cex=1.1,font=2), strip.background=list(col="grey85"))))
dev.off()
```

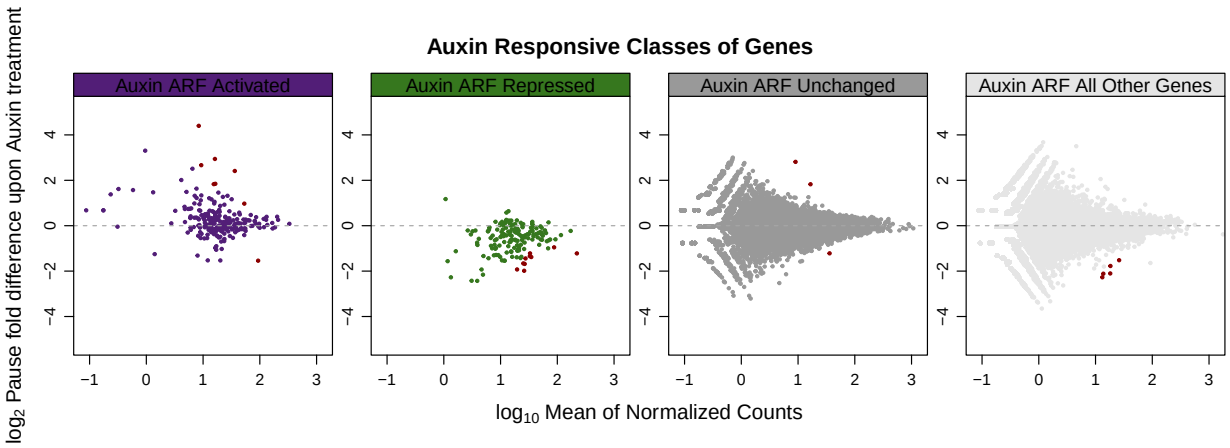

Figure 16: RNA Polymerase pause density is reduced in the repressed gene class. Although not significant (this FDR is 0.01), this is probably due to two factors: 1) we are looking in a 50 base pair window, so there are few reads associated with any given gene; and 2) calling transcription start sites and pause regions is non-trivial.

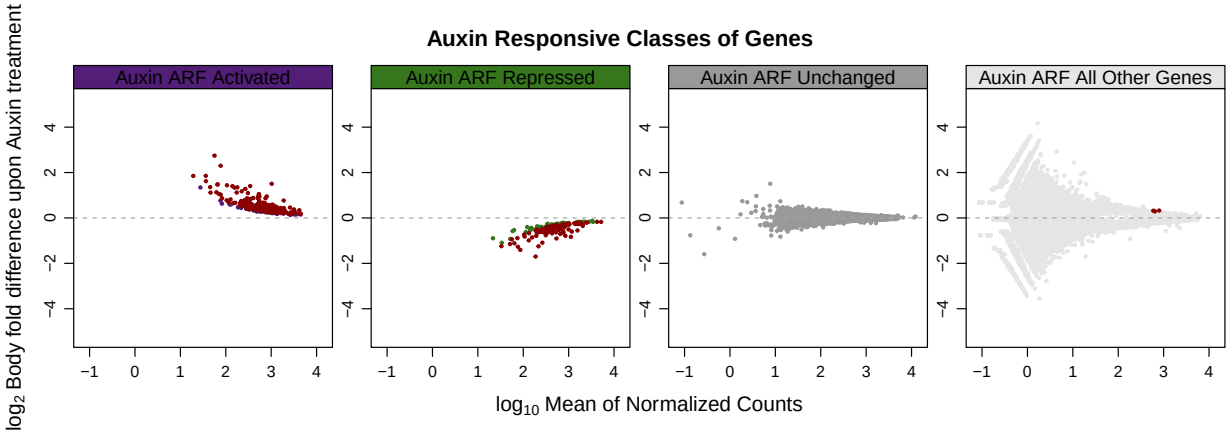

Figure 17: The changes in the gene body are consistent with how the gene classes were defined, so this just shows that changes in the pause region are not driving the original definition of gene classes.

6.4 Exhausting the pause analysis

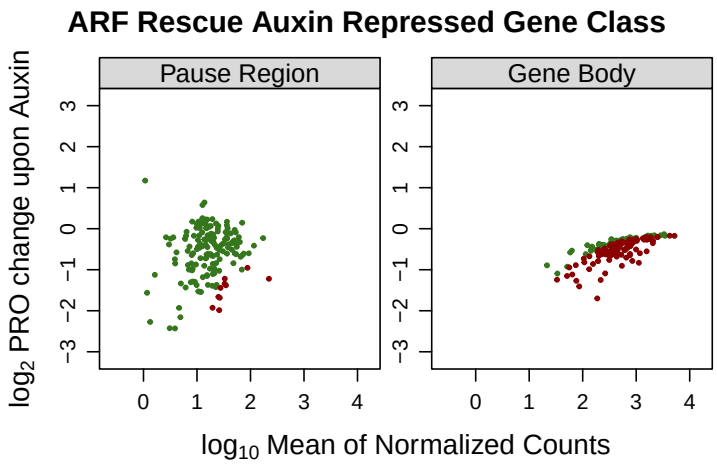

Figure 18: Focusing on the repressed class we can compare the pause and gene body changes.

```
options(scipen=999)


bed.auxin.tss.activated = filter.deseq.into.bed(df.auxin.effects, gene.file, cat = 'Auxin ARF Activated')
bed.auxin.tss.repressed = filter.deseq.into.bed(df.auxin.effects, gene.file, cat = 'Auxin ARF Repressed')
bed.auxin.tss.unchanged = filter.deseq.into.bed(df.auxin.effects, gene.file, cat = 'Auxin ARF Unchanged')
bed.auxin.tss.dregs = filter.deseq.into.bed(df.auxin.effects, gene.file, cat = 'Auxin ARF All Other Genes')



bed.auxin.tss.activated[,5] = 'Activated'
bed.auxin.tss.repressed[,5] = 'Repressed'
bed.auxin.tss.unchanged[,5] = 'Unchanged'
bed.auxin.tss.dregs[,5] = 'All Other Genes'
colnames(bed.auxin.tss.activated) = c('chr', 'start', 'end', 'gene', 'class', 'strand')
colnames(bed.auxin.tss.repressed) = c('chr', 'start', 'end', 'gene', 'class', 'strand')
colnames(bed.auxin.tss.unchanged) = c('chr', 'start', 'end', 'gene', 'class', 'strand')
colnames(bed.auxin.tss.dregs) = c('chr', 'start', 'end', 'gene', 'class', 'strand')

bed.auxin.tss = rbind(bed.auxin.tss.activated, bed.auxin.tss.repressed,
                      bed.auxin.tss.unchanged, bed.auxin.tss.dregs)


bw.plot.pause.input = bwplot.input.pause(direc, bed.auxin.tss, loaded.bw.plus,
                                         loaded.bw.minus, file.prefix = 'HEK_TIR1_ZNF143AID_ARF')
save(bw.plot.pause.input, file = paste0(direc, 'bw.plot.pause.input.Rdata'))

bw.plot.body.input = bwplot.input.body(direc, bed.auxin.tss, gene.file, file.prefix = 'HEK_TIR1_ZNF143AID_ARF')

df.bwplot.pause = data.frame(matrix(ncol = 3, nrow = nrow(bw.plot.pause.input)))
colnames(df.bwplot.pause) = c('value', 'cond', 'grp')

                                        #

df.bwplot.pause$value = bw.plot.pause.input$HEK_TIR1_ZNF143AID_ARF_Auxin/bw.plot.pause.input$HEK_TIR1_ZNF143AID_ARF

df.bwplot.pause$grp <- factor(as.character(bw.plot.pause.input$class),
                              levels = c("Activated", "Unchanged", "Repressed", "All Other Genes"))

df.bwplot.pause$cond =  'Pause Region'

                                        #ad hoc filter that should get rid of poor annotations/low expressed genes
filter = 0.01
dim(df.bwplot.pause)
#df.bwplot.pause = df.bwplot.pause[bw.plot.pause.input$HEK_TIR1_ZNF143AID_ARF_Auxin >
#filter & bw.plot.pause.input$HEK_TIR1_ZNF143AID_ARF > filter ,]
dim(df.bwplot.pause)

df.bwplot.body = data.frame(matrix(ncol = 3, nrow = nrow(bw.plot.pause.input)))
colnames(df.bwplot.body) = c('value', 'cond', 'grp')
```

```
df.bwplot.body$value = bw.plot.body.input$HEK_TIR1_ZNF143AID_ARF_Auxin/bw.plot.body.input$HEK_TIR1_ZNF143AID_ARF


df.bwplot.body$grp <- factor(as.character(bw.plot.body.input$class),
                             levels = c("Activated", "Unchanged", "Repressed", "All Other Genes"))
df.bwplot.body$cond =  'Gene Body'

#ad hoc filter that should get rid of poor annotations/low expressed genes
dim(df.bwplot.body)
#df.bwplot.body = df.bwplot.body[bw.plot.body.input$HEK_TIR1_ZNF143AID_ARF_Auxin >
#0.00001 & bw.plot.body.input$HEK_TIR1_ZNF143AID_ARF > 0.00001,]
dim(df.bwplot.body)

df.bwplot = rbind(df.bwplot.pause, df.bwplot.body)
df.bwplot$cond <- factor(as.character(df.bwplot$cond), levels = c("Pause Region", "Gene Body"))

pdf(paste0(direc, 'test_change_density_bwplot_2.pdf'), useDingbats = FALSE, width=8.83, height=3.4)
trellis.par.set(box.umbrella = list(lty = 1, col="black", lwd=2),
                box.rectangle = list(col = 'black', lwd=1.6),
                plot.symbol = list(col='black', lwd=1.6, pch ='.'))

bwplot(log(as.numeric(as.character(value)), base = 2) ~ cond |grp, data = df.bwplot,
       between=list(y=1.0, x = 1.0),
       layout=c(4,1),
       scales=list(relation="free",rot = 45, alternating=c(1,1,1,1)),
       ylab = expression('log'[2]*' Change in Pol II Density'),
       horizontal =FALSE, pch = '|', col= 'black',
       aspect = 1.5,
       index.cond = list(c(1,3,2,4)),
       ylim = c(-6, 6),
       par.settings=list(par.xlab.text=list(cex=1.0,font=1),
           par.ylab.text=list(cex=1.0,font=1),
           par.main.text=list(cex=1.0, font=1),
           plot.symbol = list(col='black', lwd=1.6, pch =19, cex = 0.25)),
       strip = function(..., which.panel, bg) {
           bg.col = c("#521e77",  "grey60",  "#36771e","grey90")
           strip.default(..., which.panel = which.panel, bg = rep(bg.col, length = which.panel)[which.panel])
       },
       panel = function(...) {
           panel.abline(h = 0, col = 'grey65')
           panel.bwplot(...)
       })
dev.off()

pdf('Change_density_bwplot_repressed.pdf',  useDingbats = FALSE, width=3.4, height=3.4)
trellis.par.set(box.umbrella = list(lty = 1, col="#36771e", lwd=2),
                box.rectangle = list(col = "#36771e", lwd=1.6),
                plot.symbol = list(col="#36771e", lwd=1.6, pch ='.'))

bwplot(log(as.numeric(as.character(value)), base = 2) ~ cond,
       data = df.bwplot[df.bwplot$grp == 'Repressed',],
       between=list(y=1.0, x = 1.0),
       #layout=c(4,1),
       scales=list(relation="free",rot = 45, alternating=c(1,1,1,1)),
       ylab = expression('log'[2]*' Change in Pol II Density'),
       horizontal =FALSE, pch = '|', col= 'black',
       aspect = 1.5,
       ylim = c(-6, 6),
       par.settings=list(par.xlab.text=list(cex=1.0,font=1),
           par.ylab.text=list(cex=1.0,font=1),
           par.main.text=list(cex=1.0, font=1),
           plot.symbol = list(col='black', lwd=1.6, pch =19, cex = 0.25)),
       strip = function(..., which.panel, bg) {
           #bg.col = c("#ce228e" ,"grey60", "#2290cf","grey90")
           strip.default(..., which.panel = which.panel,
                         bg = rep(bg.col, length = which.panel)[which.panel])
       },
       panel = function(...) {
           panel.abline(h = 0, col = 'grey65')
           panel.bwplot(...)
       })
dev.off()
```

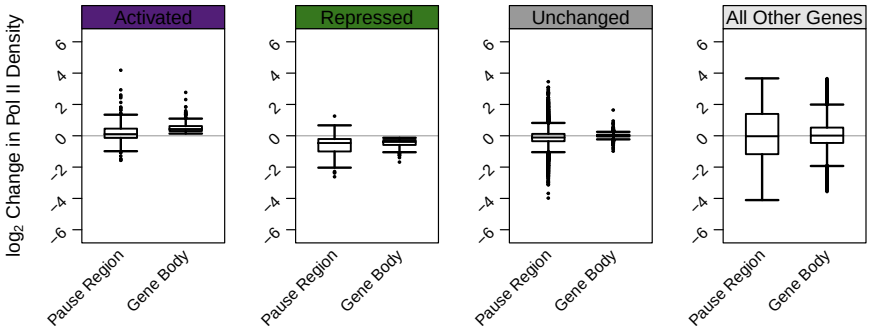

Figure 19: Pause density decreases comparably in the pause region and the gene body.

```
pi = bw.plot.pause.input$HEK_TIR1_ZNF143AID_ARF/bw.plot.body.input$HEK_TIR1_ZNF143AID_ARF
pause.index = cbind(bw.plot.pause.input, pi)

pause.index$class <- factor(as.character(pause.index$class),
                            levels = c("Activated", "Unchanged", "Repressed", "All Other Genes"))

pdf(paste0(direc, 'Pause_Index_basal_condition.pdf'), useDingbats = FALSE, width=5.83, height=4.8)
trellis.par.set(box.umbrella = list(lty = 1,
                                    col=c("#521e77",  "grey60",  "#36771e","grey90"), lwd=2),
                box.rectangle = list(col =
                                         c("#521e77",  "grey60",  "#36771e","grey90"), lwd=1.6),
                plot.symbol = list(col='black', lwd=1.6, pch ='.'))

bwplot(log(pi, base = 2) ~ class, data =pause.index,
       between=list(y=1.0, x = 1.0),
       #layout=c(4,1),
       scales=list(relation="free",rot = 45, alternating=c(1,1,1,1)),
       ylab = expression('log'[2]*'(Pause Density/Body Density)'),
       horizontal =FALSE, pch = '|', col= 'black',
       main = "Pause Index prior to Auxin treatment",
       aspect = 1.4,
       par.settings=list(par.xlab.text=list(cex=1.0,font=1),
           par.ylab.text=list(cex=1.0,font=1),
           par.main.text=list(cex=1.0, font=1),
           plot.symbol = list(col='black', lwd=1.6, pch =19, cex = 0.25)),
       strip = function(..., which.panel, bg) {
           bg.col = c("#ce228e" ,"grey60", "#2290cf","grey90")
           strip.default(..., which.panel = which.panel,
                         bg = rep(bg.col, length = which.panel)[which.panel])
       },
       panel = function(...) {
           panel.abline(h = 0, col = 'grey65')
           panel.bwplot(...)
       })
dev.off()



pi.treat = bw.plot.pause.input$HEK_TIR1_ZNF143AID_ARF_Auxin/bw.plot.body.input$HEK_TIR1_ZNF143AID_ARF_Auxin
pause.index.treat = cbind(bw.plot.pause.input, pi.treat)

pause.index.treat$class <- factor(as.character(pause.index.treat$class),
                            levels = c("Activated", "Unchanged", "Repressed", "All Other Genes"))

pdf(paste0(direc, 'Pause_Index_auxin_condition.pdf'), useDingbats = FALSE, width=5.83, height=4.8)
trellis.par.set(box.umbrella = list(lty = 1,
                                    col=c("#521e77",  "grey60",  "#36771e","grey90"), lwd=2),
                box.rectangle = list(col =
                                         c("#521e77",  "grey60",  "#36771e","grey90"), lwd=1.6),
                plot.symbol = list(col='black', lwd=1.6, pch ='.'))

bwplot(log(pi.treat, base = 2) ~ class, data =pause.index.treat,#~ dtx | class), data =pause.index,
       between=list(y=1.0, x = 1.0),
       #layout=c(4,1),
       scales=list(relation="free",rot = 45, alternating=c(1,1,1,1)),
       ylab = expression('log'[2]*'(Pause Density/Body Density)'),
       horizontal =FALSE, pch = '|', col= 'black',
       main = "Pause Index upon Auxin treatment",
       aspect = 1.4,
       par.settings=list(par.xlab.text=list(cex=1.0,font=1),
           par.ylab.text=list(cex=1.0,font=1),
           par.main.text=list(cex=1.0, font=1),
```

```
            plot.symbol = list(col='black', lwd=1.6, pch =19, cex = 0.25)),
       strip = function(..., which.panel, bg) {
            bg.col = c("#ce228e" ,"grey60", "#2290cf","grey90")
            strip.default(..., which.panel = which.panel,
                          bg = rep(bg.col, length = which.panel)[which.panel])
       },
       panel = function(...) {
            panel.abline(h = 0, col = 'grey65')
            panel.bwplot(...)
       })
dev.off()


#according to the model pause index should not change.

pause.index.change = cbind(bw.plot.pause.input, pi.treat, pi)

pause.index.change$class <- factor(as.character(pause.index.change$class),
                              levels = c("Activated", "Unchanged", "Repressed", "All Other Genes"))

pdf(paste0(direc, 'Pause_Index_change_condition.pdf'), useDingbats = FALSE, width=5.83, height=4.8)
trellis.par.set(box.umbrella = list(lty = 1,
                                    col=c("#521e77",  "grey60",  "#36771e","grey90"), lwd=2),
                box.rectangle =
                    list(col = c("#521e77",  "grey60",  "#36771e","grey90"), lwd=1.6),
                plot.symbol = list(col='black', lwd=1.6, pch ='.'))

bwplot(log(pi.treat/pi, base = 2) ~ class, data =pause.index.change,
       between=list(y=1.0, x = 1.0),
       #layout=c(4,1),
       scales=list(relation="free",rot = 45, alternating=c(1,1,1,1)),
       ylab = expression('log'[2]*'(change in Pause Index upon Auxin)'),
       horizontal =FALSE, pch = '|', col= 'black',
       aspect = 1.4,
       par.settings=list(par.xlab.text=list(cex=1.0,font=1),
            par.ylab.text=list(cex=1.0,font=1),
            par.main.text=list(cex=1.0, font=1),
            plot.symbol = list(col='black', lwd=1.6, pch =19, cex = 0.25)),
       strip = function(..., which.panel, bg) {
            bg.col = c("#ce228e" ,"grey60", "#2290cf","grey90")
            strip.default(..., which.panel = which.panel,
                          bg = rep(bg.col, length = which.panel)[which.panel])
       },
       panel = function(...) {
            panel.abline(h = 0, col = 'grey65')
            panel.bwplot(...)
       })
dev.off()
```

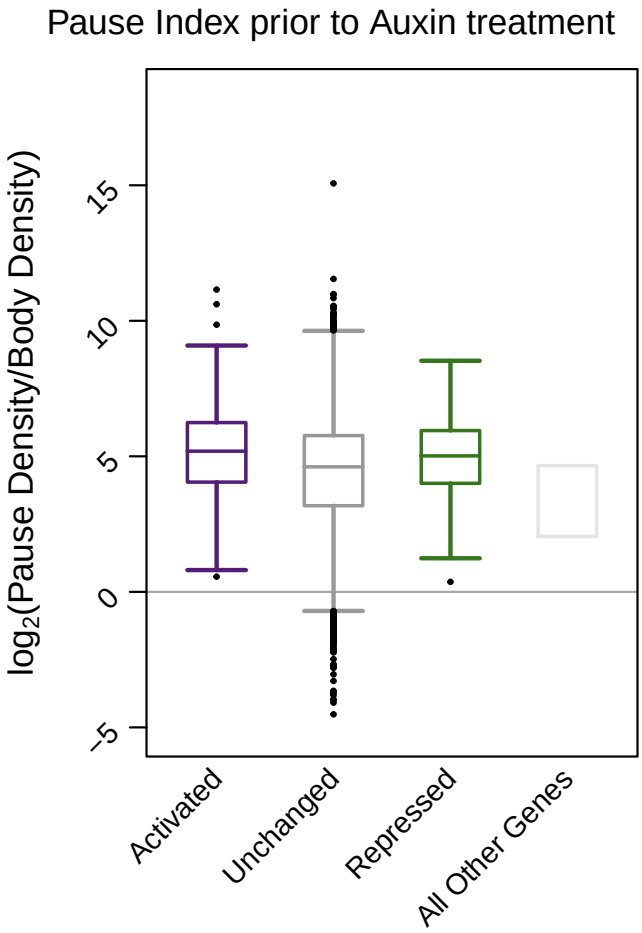

Figure 20: Genes in the activated, repressed, and unchanged classes are all highly paused prior to auxin treatment.

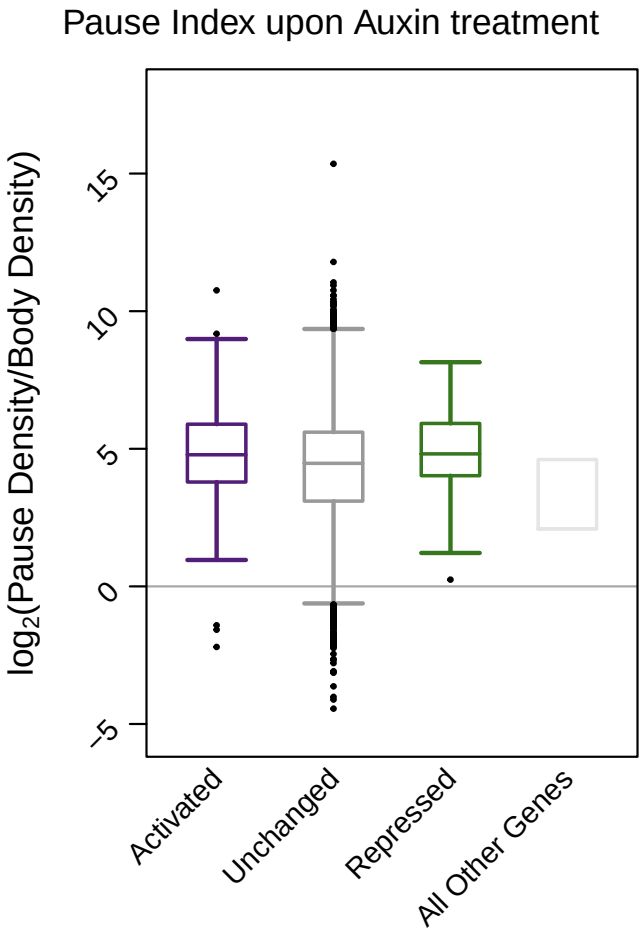

Figure 21: Genes in the activated, repressed, and unchanged classes remain paused after auxin treatment.

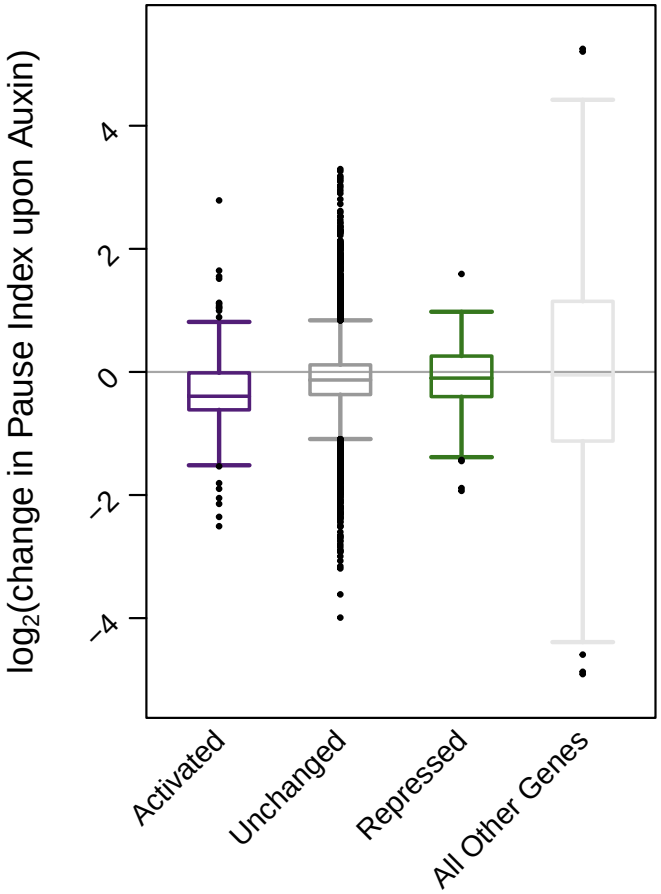

Figure 22: The repressed gene class has a comparable change in pause index to the unchanged class. Both of these changes are modest. The activated gene class has a more substantial effect, which is driven by the relatively modest change in the pause region and a consistent increase throughout the gene body (Figure 15).

7 Modeling changes in RNA Polymerase density

To better understand the potential mechanisms underlying decreased pause/body densities for repressed genes following ZNF143 inhibition, we implemented a mathematical modeling approach. We formulated a simple two compartment model with dynamics for a pause region and gene body region. Model parameter included rate constants for transcriptional initiation, premature pause release, pause release into transcriptional elongation, and termination of transcription (Figure 23A,B). We implemented parameter sensitivity analyses to determine if any model parameters could account for our experimental results. Our analysis showed that only changes in initiation and premature release could account for our experimental results. Increased initiation resulted in increases for both the pause region and the gene body region. In contrast, increases in premature release resulted in decreases for both the pause region and the gene body region (Figure 23C). These results suggest that decreases in the pause and gene body regions observed following ZNF143 inhibition could be accounted for by either decreases in initiation or increases in premature release, both of which are upstream of pausing. We used the model to further investigate the conditions under which the change in the pause region could be greater than the change in the gene body region for a variation in either the initiation or premature release rate. Our analysis showed that such preferential changes in the pause region will always be observed when the pause index is greater than unity (i.e., there is greater polymerase density in the pause region as compared to the gene body region). Moreover, the relative pause change was predicted by the model to be identical to the pause index (Figure 23D). The combined experimental and modeling results suggest that ZNF143 augments transcription through mechanisms upstream of polymerase pausing and the extent of its preferential effects on pausing are directly related to the relative degree of pausing.

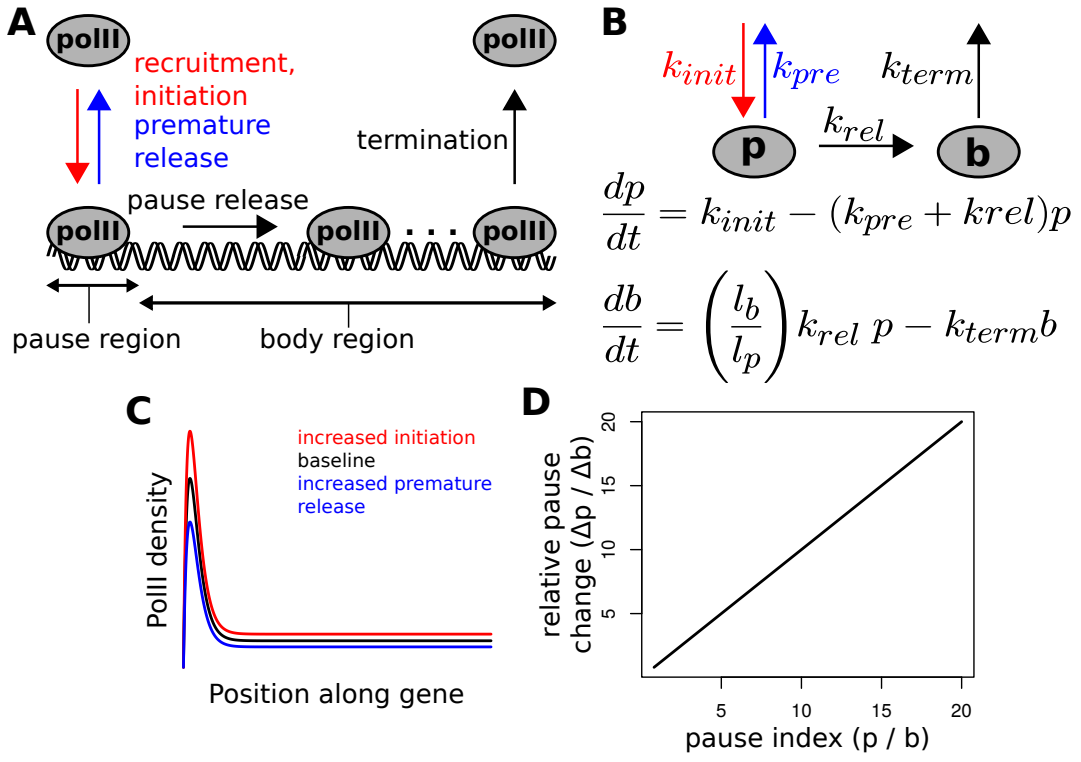

Figure 23: ZNF143 regulates transcription through influences on RNA Polymerase recruitment/initiation or premature pause release. (A) Schematic illustrating model structure and key variables. (B) Reduced schematic with mathematical formulation of the two component model in which p refers to the polymerase density at the pause region and b refers to the density at the gene body region. (C) Steady state simulation for a reference model (black), a model in which inscriptional initiation was increased by 25% (red), and a model in which premature pause release was increased by 60% (blue). The peak of the smooth curve was set to the steady state pause level and the plateau of the curve was set to the steady state gene body level. Note the preferential effect on the pause region as compared to the gene body region. (D) The change in the pause region divided by the change in the gene body region directly related to the pause index, as determined by model analysis.

7.1 Model formulation

The dynamics for the concentrations or densities of polymerases at pausing regions and gene bodies, defined as p and b, are described as follows:

$$\frac{dp}{dt} = k_{init} - (k_{pre} + k_{rel})\ p$$
$$\frac{db}{dt} = \left(\frac{l_p}{l_b}\right) k_{rel}\ p - k_{term}\ b$$

where k_{init} is the rate of transcription initiation, k_{pre} is the rate constant for premature relase of a paused polymerase, k_{rel} is the rate constant for the release of a paused polymerase as transcription procedes into the gene body, and k_{term} is the rate constant for the termination of transcription. The term l_p/l_b is a ratio of the relative DNA segment lengths that is applied to adjust the gene body concentration based on the larger amount of DNA in the gene body as compared to the pause region. The term k_{init} implicitly accounts for the product of the unbound polymerase concentration and the rate constant for initiation. Concentration is considered as number of polymerases per length of DNA. Thus this concentration is often referred to as a density. We consider this model as through the units are dimensionless for the analysis of how specific rates influence the relative polymerase quantitaties at pause sites and gene bodies. The steady state levels (p_{ss}, b_{ss}) are found by setting $dp/dt = db/dt = 0$. Because understanding the effects of the relative pause region length is not our focus, we set $r = l_p/l_b$ for our parameter sensitivity analyses.

$$p_{ss} = \frac{k_{init}}{k_{pre} + k_{rel}}$$
$$b_{ss} = \frac{r\ k_{rel}}{k_{term}}\ p_{ss} = \frac{r\ k_{rel}\ k_{init}}{k_{term}\ (k_{pre} + k_{rel})}$$

Thus, the pause index or relative density of reads in the pause region ($P_i = p_{ss}/b_{ss}$) is only dependent on k_{term} and k_{rel}, the rate constants for the termination of transcription and pause release:

$$P_i = \frac{k_{term}}{r\ k_{rel}}$$

7.2 Parameter sensitivity analysis

Here we document the effects of each parameter on the pause and body concentrations by considering high and low parameter values ($k^{(hi)}$ and $k^{(lo)}$) and computing the changes in p_{ss} and b_{ss}. We use the following notation to document the change in p_{ss} when k is changed from a relatively high to a relatively low value: $\Delta p(k) = p_{ss}(k^{(hi)}) - p_{ss}(k^{(lo)})$). First, we evaluate the effects of the transcription initiation rate (k_{init}) on the steady state pause region and gene body polII concentrations.

$$\Delta p(k_{init}) = \frac{k_{init}^{(hi)} - k_{init}^{(lo)}}{k_{pre} + k_{rel}}$$
$$\Delta b(k_{init}) = \frac{r\ k_{rel}\ \left(k_{init}^{(hi)} - k_{init}^{(lo)}\right)}{k_{term}\ (k_{pre} + k_{rel})}$$

Note that $k^{(hi)} > k^{(lo)}$ by definition, so that $\Delta p(k_{init}) > 0$ and $\Delta b(k_{init}) > 0$. Therefore, increasing the rate of transcription initiation will result in both pause region and gene body increases. We also note that we are presenting results of single parameter changes that can also be considered using a standard sensitivity analysis. For this example the sensitivities are computed as follows:

$$\frac{\partial p_{ss}}{\partial k_{init}} = \frac{1}{k_{pre} + k_{rel}} > 0$$
$$\frac{\partial b_{ss}}{\partial k_{init}} = \frac{r\ k_{rel}}{k_{term}\ (k_{pre} + k_{rel})} > 0$$

In general, $\Delta p(k) \sim (\partial p/\partial k)(k^{(hi)} - k^{(lo)})$ for small changes in k and $sign(\Delta p(k)) = sign(\partial p/\partial k)$. Next we consider the effects of varying the rate constant for premature pause release (k_{pre}).

$$\Delta p(k_{pre}) = \frac{k_{init}\left(k_{pre}^{(lo)} - k_{pre}^{(hi)}\right)}{\left(k_{pre}^{(hi)} + k_{rel}\right)\left(k_{pre}^{(lo)} + k_{rel}\right)}$$

Table 1: **Parameter sensitivity summary**

Parameter increased	**Change in p_{ss}**	**Change in b_{ss}**
k_{init}	increase	increase
k_{pre}	decrease	decrease
k_{rel}	decrease	increase
k_{term}	no change	decrease

$$\Delta b(k_{pre}) = \frac{r\ k_{rel}\ k_{init}\left(k_{pre}^{(lo)} - k_{pre}^{(hi)}\right)}{k_{term}\left(k_{pre}^{(hi)} + k_{rel}\right)\left(k_{pre}^{(lo)} + k_{rel}\right)}$$

These results show that increasing the rate constant for premature pause release will decrease both pause region and gene body polymerase concentrations ($\Delta p(k_{pre}) < 0$ and $\Delta b(k_{pre}) < 0$). We next demonstrate that an increase in the rate constant for the release of polymerases into the gene body (k_{rel}) decreases p_{ss} and increases b_{ss}:

$$\Delta p(k_{rel}) = \frac{k_{init}\left(k_{rel}^{(lo)} - k_{rel}^{(hi)}\right)}{\left(k_{pre} + k_{rel}^{(hi)}\right)\left(k_{pre} + k_{rel}^{(lo)}\right)}$$

$$\Delta b(k_{rel}) = \frac{r\ k_{init}\ k_{pre}\left(k_{rel}^{(hi)} - k_{rel}^{(lo)}\right)}{k_{term}\left(k_{pre} + k_{rel}^{(hi)}\right)\left(k_{pre} + k_{rel}^{(lo)}\right)}$$

It is interesting to note that if there is no premature pause release (i.e., $k_{pre} = 0$), the model predicts that that a change in the rate of pause release into transcriptional elongation will not affect the concentration of polymerases in the gene body, if all other factors are identical (i.e., $\Delta b(k_{rel}) = 0$ for $k_{pre} = 0$). Finally, we evaluate the effect of modifying the rate constant for the termination of transcription:

$$\Delta p(k_{term}) = 0$$

$$\Delta b(k_{term}) = \frac{r\ k_{init}\ k_{pre}\left(k_{term}^{(lo)} - k_{term}^{(hi)}\right)}{k_{term}^{(hi)} k_{term}^{(lo)}\left(k_{pre} + k_{rel}\right)}$$

These results shows that an increase in the rate constant for transcriptional termination will no affect the steady state level of paused polymerase but will decrease the gene body concentration of polymerases ($\Delta p(k_{term}) = 0$, $\Delta b(k_{term}) < 0$). The results from our parameter sensitivity analyses are illustrated in Table 1 where the entries for p_{ss} and b_{ss} indicate the effect of increasing each parameter.

7.3 Relative effects of transcriptional initiation and premature release on the distribution of polymerases in the pause region and gene body

Our preceding analyses show that consistent changes in the pause region and gene body (i.e., both increase or both decrease) are only observed following changes in the rate of transcriptional initiation or the rate constant for premature pause relase (k_{init}, k_{rel}). Experimental data show that repressed genes following transcription factor inhibition show decreases in both the pause region and the gene body region. These decreases were greater in the pause region in comparison to the gene body. Here we document the conditions under which the effects of varying the rates of initiation and premature release of the paused polymerase are greater for the pause region as compared to the gene body (e.g., $\Delta p(k_{init}) > \Delta b(k_{init})$). This conditon is as follows changes in transcriptional initiation, where we set $k_{init}^{(hi)} - k_{init}^{(lo)} = \Delta k$:

$$\frac{\Delta k}{k_{pre} + k_{rel}} > \frac{r\ k_{rel}\ \Delta k}{k_{term}\ (k_{pre} + k_{rel})}$$

This condition is satisfied for $\frac{l_b}{l_p} > \frac{k_{rel}}{k_{term}}$ (recall $r = l_p/l_b$). In general, l_b/l_p is a large value because the pause region (approximately <100 bp) is much smaller than the gene body (approximately >10 kb). So $k_{rel}/k_{term} < l_b/l_p \sim 100$ must be obtained. For changes in premature relase, the condition is as follows:

$$\frac{k_{init}(-\Delta k)}{\left(k_{pre}^{(hi)}+k_{rel}\right)\left(k_{pre}^{(lo)}+k_{rel}\right)} > \frac{r\ k_{rel}\ k_{init}(-\Delta k)}{k_{term}\left(k_{pre}^{(hi)}+k_{rel}\right)\left(k_{pre}^{(lo)}+k_{rel}\right)}$$

Again, this relation leads to the same constraint observed for k_{init}: $k_{rel}/k_{term} < l_b/l_p$. Recall that the pause index is defined as $P_i = k_{term}/r\ k_{rel} = (l_b/l_p)(k_{term}/k_{rel})$. Therefore, $P_i > 1$ for the same condition that constrains $\Delta p(k_{pre}) > \Delta b(k_{pre})$ for changes in k_{init} and k_{pre}: $k_{rel}/k_{term} < l_b/l_p$. This demonstrates that, for an arbitrary change in either the initiation rate or the premature pause relase rate constant, the effect of the change in the pause region will be greater than the change in the gene body region whenever the pause region polymerase concentration is greater than that at the gene body. Further, the effect ratio is identical for changes in k_{init} and k_{rel} and is equal to the value of the pause index: $\Delta p/\Delta b = P_i$.

7.4 Pause region and gene body visualization

We aimed to generate plots in which steady state levels of pause region and gene body concentration were imposed upon the peak and flat regions of a profile that is characterized by an exponential approach towards the peak followed by an exponential decay towards a stable palteau. We used a sum of exponential functions for the waveform:

$$density(bp) = \frac{pk_{pause}}{pk}\left(\frac{bp}{\tau}exp\left(\frac{-(bp-\tau)}{\tau}\right) + pk_{body}\left(1-exp\left(\frac{-bp}{\tau}\right)\right)\right)$$

where bp (base pairs) is the independent variable and τ is the exponential decay constant. The parameter pk is set to the root of the derivative as shown below so that pk_{pause} determines the peak of the waveform. The parameter pk_{body} is set such that the assymptotic gene body region decays to a desired level. The derivative of this waveform is

$$\frac{d}{dt}density(bp) = \frac{pk_{pause}}{pk}\left(\frac{pk_{body}}{\tau}e^{-bp/\tau} - \frac{bp}{\tau^2}e^{(\tau-bp)/\tau} + \frac{1}{\tau}e^{(\tau-bp)/\tau}\right)$$

and the root of the derivative is given by the value of bp at the peak of the waveform ($max(density)$):

$$bp = \frac{\tau(pk_{body}+e)}{e}$$

For $bp >> \tau$, the gene body level assymptotically approaches $pk_{body}\ pk_{pause}/pk$. We implicitly determine a value for pk_{body} that will give a gene body level of a desired level as follows using R. That is, we select a value of pk_{body} that will give a plateau $pk_{body}\ pk_{pause}/pk$ of choice for a given setting of the pause peak pk_{pause}:

```
# function for finding the gene body parameter
# bpeak: desired gene body level
# pausepeak: desired pause region level
# bp.seq: base pair sequence
# tau: exponential decau constant
# min.pk: minimal level for gene body peak
# max.pk: maximal level for gene body peak
# dpk: resolution for the implicit solution
find.body.param = function(bpeak=NULL,bp.seq=NULL,tau=NULL,
                           pausepeak=NULL,min.pk=0,max.pk=1,
                           dpk=0.001) {
  pk = seq(min.pk,max.pk,dpk)  # sequence of peak parameter values
  bp = bp.seq                  # base pairs, x-axis
  dat = matrix(0,length(pk),2) # data matrix
  colnames(dat) = c("body_param","body_assymp")
  for(x in 1:length(pk)) {
    bodypeak = pk[x]
    root = tau*(bodypeak+exp(1))*exp(-1)
    peak = (root/tau) * exp(-(root - tau)/tau) + bodypeak *
      (1 - exp(-root/tau))
    body = bodypeak * pausepeak / peak
    dat[x,1] = bodypeak
    dat[x,2] = body
  }
  # look up the ratio of the body parameter over the assymptote
  # use linear interpolation
  inter = findInterval(bpeak,dat[,2])
  m = (dat[(inter+1),1] - dat[inter,1]) /
    (dat[(inter+1),2] - dat[inter,2])
  b = dat[inter,1] - m * dat[inter,2]
  est = m * bpeak + b
  return(est)
} # find.body.param
```

After implicitly finding a value for pk_{body} that produces the gene body level of choice, the waveform is produced with the following function:

```
# function to get the PRO waveform
# bpeak: desired gene body level
# pausepeak: desired pause region level
# bodypeak: body peak value to obtain a level of bpeak
# bp.seq: base pair sequence
# tau: exponential decau constant
get.pro.waveform = function(bpeak=NULL,pausepeak=NULL,bodypeak=NULL,
                            bp.seq=NULL,tau=NULL){
  root = tau*(bodypeak+exp(1))*exp(-1)
  peak = (root/tau) * exp(-(root - tau)/tau) + bodypeak *
    (1 - exp(-root/tau))
  vec =  sapply(bp.seq,function(x){(pausepeak/peak)*((x/tau) *
         exp(-(x - tau)/tau) + bodypeak * (1 - exp(-x/tau)))})
  vec = unname(vec)
  pars = as.data.frame(rbind(cbind(max(vec),
       vec[bp.seq[length(bp.seq)]]),c(pausepeak,bpeak)))
  names(pars) = c("Peak","Assymp")
  rownames(pars) = c("simulated","requested")
  out = list(vec=vec, pars=pars)
  return(out)
} # get.pro.waveform
```

Note that the output of this function allows the user to evaluate the accuracy of the implicit solution for pk_{body}. Here is an example in which the code outputs a waveform with a pause region level of 2 ($pausepeak = 2$) and a gene body level of 1 ($bpeak = 1$; Figure 23).

```
delta.bp = 1000
tau=20
bp = seq(0,delta.bp-1)
pausepeak = 2
bpeak = 1
bodypeak = find.body.param(bpeak=bpeak,bp.seq=bp,tau=tau,
             pausepeak=pausepeak,min.pk=0,max.pk=2,dpk=0.001)
vec = get.pro.waveform(pausepeak=pausepeak,bpeak=bpeak,
                       bodypeak=bodypeak,bp.seq=bp,tau=tau)

pdf(paste0(direc, 'model_plot.pdf'), width=5.83, height=4.8)

plot(bp,vec[[1]],type="n",xlab="chrom coords",ylab="reads")
lines(bp,vec[[1]])
dev.off()
```

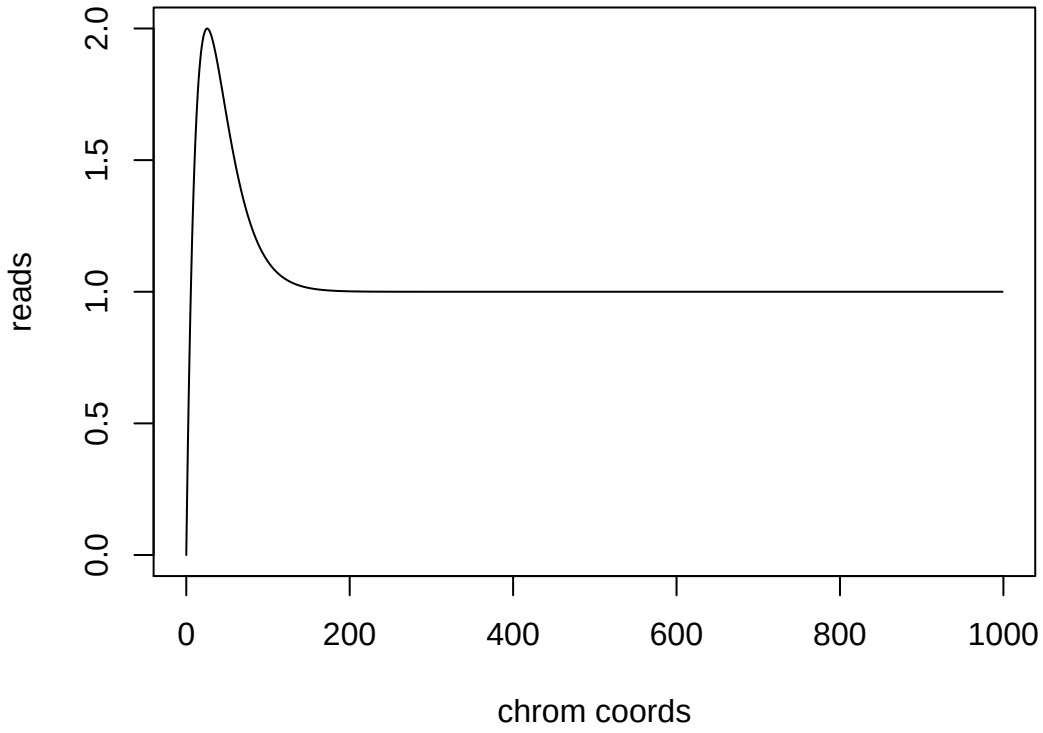

Figure 24: Plotting pause region and gene body levels with a smooth function.

7.5 Final modeling figure code

The following code was used to generate the plots shown in the paper.

```
#################################################################
## results for changes in transcription initiation and premature relase
#################################################################

# sim params
delta.bp = 1000
tau=20
bp = seq(0,delta.bp-1)

# baseline model
lp = 100
lb = 28464
kinit = 1 * (2000 / lb)
krel = 1 * (2000 / lb)
kpre = 1 * (2000 / lb)
kterm = 0.025 * (2000 / lb)
p = kinit / (krel + kpre)
b = (lp / lb) * (krel / kterm) * p
bodypeak = find.body.param(bpeak=b, bp.seq=bp,tau=tau,
          pausepeak=p, min.pk=0, max.pk=1, dpk=0.001)
vec = get.pro.waveform(pausepeak=p, bpeak=b,
            bodypeak=bodypeak, bp.seq=bp, tau=tau)
plot(bp,vec[[1]],type="n",xlab="chrom coords",ylab="reads",ylim=c(0,0.6))
lines(bp,vec[[1]],cex=2)
baseline = vec[[1]]

# increase kinit
kpre = 1 * (2000 / lb)
kinit = 1.25 * (2000 / lb)
p = kinit / (krel + kpre)
b = (lp / lb) * (krel / kterm) * p
bodypeak = find.body.param(bpeak=b, bp.seq=bp,tau=tau,
                           pausepeak=p, min.pk=0, max.pk=1, dpk=0.001)
vec = get.pro.waveform(pausepeak=p, bpeak=b,
                       bodypeak=bodypeak, bp.seq=bp, tau=tau)
plot(bp,baseline,type="n",xlab="chrom coords",ylab="reads",ylim=c(0,0.6))
lines(bp,baseline,cex=2)
lines(bp,vec[[1]],cex=2,col="red")
kinit.up = vec[[1]]

# increase kpre
kinit = 1 * (2000 / lb)
kpre = 1.6 * (2000 / lb)
p = kinit / (krel + kpre)
b = (lp / lb) * (krel / kterm) * p
bodypeak = find.body.param(bpeak=b, bp.seq=bp,tau=tau,
                           pausepeak=p, min.pk=0, max.pk=1, dpk=0.001)
vec = get.pro.waveform(pausepeak=p, bpeak=b,
                       bodypeak=bodypeak, bp.seq=bp, tau=tau)
plot(bp,baseline,type="n",xlab="chrom coords",ylab="reads",ylim=c(0,0.6))
lines(bp,baseline,cex=2)
lines(bp,vec[[1]],cex=2,col="red")
kpre.up = vec[[1]]

# plot all three changes
pdf(paste0(direc, 'pausedat.pdf'), width=6.83, height=3.8)
par(mfrow=c(1,2), pty="s")
lwd=2
plot(bp,baseline,type="n",xlab="chrom coords",ylab="reads", ylim=c(0,0.65))
lines(bp,baseline,lwd=lwd,col="black")
lines(bp,kinit.up,lwd=lwd,col="red")
lines(bp,kpre.up,lwd=lwd,col="blue")

# plot change versus pause index
pi = seq(0.8,20,0.2)
plot(pi,pi, type="n",ylab="change over change",xlab="pi")
lines(pi,pi,lwd=lwd)
dev.off()
```

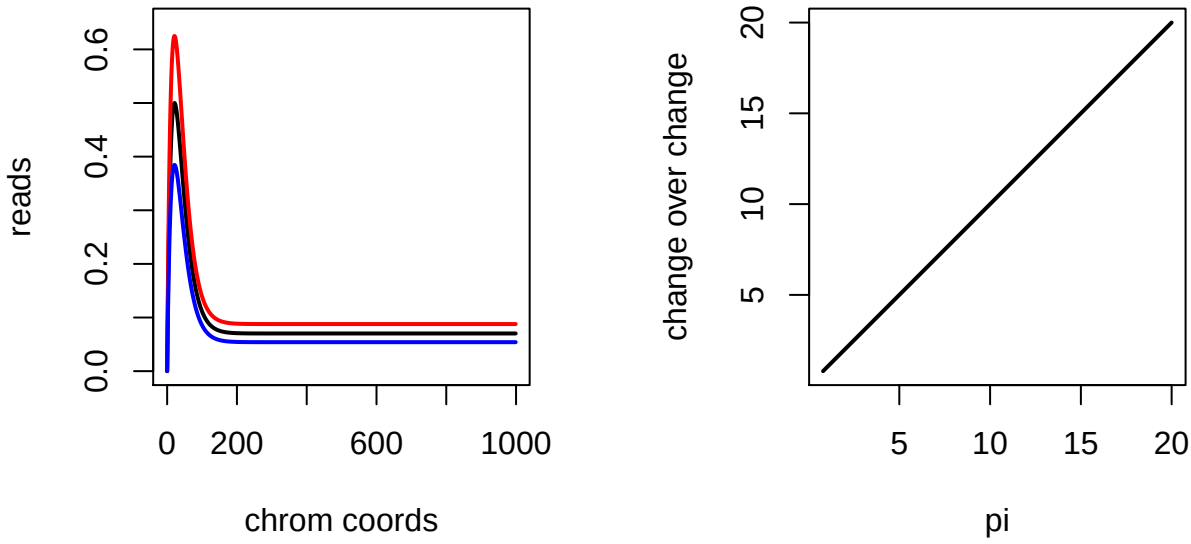

Figure 25: We set the rate of transcription initiation, the rate constant for premature release, and the rate constant for relase from pausing into transcription to an identical level. We set the rate constant for termination of transcription to a relatively small level. Plots are for steady state levels of pause reregion and gene body density or concentration.

References

Bailey TL, Boden M, Buske FA, Frith M, Grant CE, Clementi L, Ren J, Li WW, Noble WS (2009). "MEME SUITE: tools for motif discovery and searching." *Nucleic acids research*, p. gkp335.

Consortium EP, *et al.* (2012). "An integrated encyclopedia of DNA elements in the human genome." *Nature*, **489**(7414), 57.

Heinz S, Benner C, Spann N, Bertolino E, Lin YC, Laslo P, Cheng JX, Murre C, Singh H, Glass CK (2010). "Simple combinations of lineage-determining transcription factors prime cis-regulatory elements required for macrophage and B cell identities." *Molecular cell*, **38**(4), 576–589.

Kuleshov MV, Jones MR, Rouillard AD, Fernandez NF, Duan Q, Wang Z, Koplev S, Jenkins SL, Jagodnik KM, Lachmann A, *et al.* (2016). "Enrichr: a comprehensive gene set enrichment analysis web server 2016 update." *Nucleic acids research*, **44**(W1), W90–W97.

Langmead B, Trapnell C, Pop M, Salzberg S (2009). "Ultrafast and memory-efficient alignment of short DNA sequences to the human genome." *Genome Biology*, **10**(3), R25. ISSN 1465-6906. doi: 10.1186/gb-2009-10-3-r25. URL `http://genomebiology.com/2009/10/3/R25`.

Li H, Handsaker B, Wysoker A, Fennell T, Ruan J, Homer N, Marth G, Abecasis G, Durbin R, *et al.* (2009). "The sequence alignment/map format and SAMtools." *Bioinformatics*, **25**(16), 2078–2079.

Lo R, Matthews J (2012). "High-resolution genome-wide mapping of AHR and ARNT binding sites by ChIP-Seq." *Toxicological sciences*, **130**(2), 349–361.

Love MI, Huber W, Anders S (2014). "Moderated estimation of fold change and dispersion for RNA-Seq data with DESeq2." *bioRxiv*.

Mahat DB, Kwak H, Booth GT, Jonkers IH, Danko CG, Patel RK, Waters CT, Munson K, Core LJ, Lis JT (2016). "Base-pair-resolution genome-wide mapping of active RNA polymerases using precision nuclear run-on (PRO-seq)." *Nature protocols*, **11**(8), 1455.

Martin M (2011). "Cutadapt removes adapter sequences from high-throughput sequencing reads." *EMBnet. journal*, **17**(1), pp–10.

Martins AL, Walavalkar NM, Anderson WD, Zang C, Guertin MJ (2018). "Universal correction of enzymatic sequence bias reveals molecular signatures of protein/DNA interactions." *Nucleic acids research*.

Reddy TE, Pauli F, Sprouse RO, Neff NF, Newberry KM, Garabedian MJ, Myers RM (2009). "Genomic determination of the glucocorticoid response reveals unexpected mechanisms of gene regulation." *Genome research*.

Wang Z, Chu T, Choate LA, Danko CG (2018). "Identification of regulatory elements from nascent transcription using dREG." *bioRxiv*, p. 321539.

Zhang Y, Liu T, Meyer CA, Eeckhoute J, Johnson DS, Bernstein BE, Nusbaum C, Myers RM, Brown M, Li W, *et al.* (2008). "Model-based analysis of ChIP-Seq (MACS)." *Genome Biol*, **9**(9), R137.