

## promoter\_comparison

### Promoters are upstream regions of all protein-coding genes

```
library(biomaRt)
mart = getParamart()

## Database connected
## biomaRt      ...      parasite_mart
## host        ...      https://parasite.wormbase.org:443/biomaRt/martservice
## dataset     ...      wbps_gene

UPSTREAM=1000
DOWNSTREAM=200
promoters = getCElegansPromoters(mart, upstream = UPSTREAM, downstream = DOWNSTREAM)

## getBM(filter = c("biotype", "species_id_1010"), value = list(
##     biotype = "protein_coding", species_id_1010 = "caelegprjna13758"),
##     attributes = c("wbps_gene_id", "external_gene_id", "chromosome_name",
##     "start_position", "end_position", "strand"))

promoters = trim(sort(promoters, ignore.strand=T)) # trim because one interval is chrIV:-359-840 at -10
head(promoters)

## GRanges object with 6 ranges and 2 metadata columns:
##      seqnames      ranges strand |   wbps_gene_id external_gene_id
##      <Rle>      <IRanges> <Rle> |   <character>      <character>
## [1]      chrI 10031-11230      - | WBGene00022277      homt-1
## [2]      chrI 10495-11694      + | WBGene00022276      nlp-40
## [3]      chrI 26582-27781      - | WBGene00022278      rcor-1
## [4]      chrI 32951-34150      - | WBGene00022279      sesn-1
## [5]      chrI 42733-43932      + | WBGene00022275      txt-7
## [6]      chrI 46461-47660      + | WBGene00044345      Y48G1C.12
## -----
##      seqinfo: 7 sequences (1 circular) from cell genome

selfOverlaps = findOverlaps(promoters, ignore.strand=T)
#head(selfOverlaps)

# selfOverlaps includes everything against itself + overlaps between promoters
# Filter out the self hits, and retain the "between" hits as "collisions".
collisions = selfOverlaps[!isSelfHit(selfOverlaps)]

overlappingPromoterRows = unique(c( from(collisions), to(collisions)))
length(overlappingPromoterRows)

## [1] 6749

sprintf("There are %d overlaps between %d promoters.", length(collisions), length(overlappingPromoterRows))

## [1] "There are 8008 overlaps between 6749 promoters."
```

```

filtered.promoters = promoters[-overlappingPromoterRows]
filtered.promoters = filtered.promoters[-which(seqnames(filtered.promoters) == 'chrM')]
sprintf("There are %d unambiguous promoters.", length(filtered.promoters))

```

```
## [1] "There are 13246 unambiguous promoters."
```

```

# -500,+200
# "There are 4256 overlaps between 4067 promoters."
# "There are 15922 unambiguous promoters."

# -1000,+200
#"There are 8008 overlaps between 6749 promoters."
#"There are 13246 unambiguous promoters."

```

```

OUTPUT_03 = normalizePath("../03_output")
PROMOTOR_BED_PATH = sprintf("%s/filtered.promoters.minus%d_plus%d.bed", OUTPUT_03, UPSTREAM, DOWNSTREAM)
write.table(filtered.promoters, PROMOTOR_BED_PATH, sep="\t", quote=F, row.names=F, col.names=F)

```

## Setup a conda environment in your shell

I had to call my local setup script .zshrc, where I have initialized conda, to have access to the “base” environment, where I have installed wiggletools and ucsc user apps.

```
$ wiggletools apply_paste filtered.promoters.minus1000_plus200.df meanI maxI filtered.promoters.minus1000_plus200.df
ELT2_LE_combined_subtracted.bw
```

The same can be done for the IDR peaks.

```

$ wiggletools apply_paste LE_IDR_peaks.df meanI maxI ELT2_LE_combined.IDR.bed ELT2_LE_combined_subtracted.IDR.bed
PROMOTOR_DF_PATH = sprintf("%s/filtered.promoters.minus%d_plus%d.df", OUTPUT_03, UPSTREAM, DOWNSTREAM)
promoters.agg = read.table(PROMOTOR_DF_PATH)
colnames(promoters.agg) <- c("chrom", "start","end", "len", "strand", "wbps_gene_id", "gene_name", "chip_signal_mean", "chip_signal_max")

IDR_peaks.agg = read.table(file.path(OUTPUT_03,"LE_IDR_peaks.df"))
IDR_peaks.agg$V4 = NULL
IDR_peaks.agg$V5 = NULL
IDR_peaks.agg$V6 = NULL
IDR_peaks.agg$V8 = NULL
colnames(IDR_peaks.agg) <- c("chrom", "start","end","intensity","nlogq","offset","signal.mean","signal.max")

gr.IDR = makeGRangesFromDataFrame(IDR_peaks.agg,keep.extra.columns = T)
seqinfo(gr.IDR) <- Seqinfo(genome="ce11")

gr.promoters = makeGRangesFromDataFrame(promoters.agg,keep.extra.columns = T)
seqinfo(gr.promoters) <- Seqinfo(genome="ce11")

```

```

chipmean.minval = min(gr.promoters$chip_signal_mean,na.rm=T)
chipmean.minval

```

```
## [1] -100.4667
```

```

chipmax.minval = min(gr.promoters$chip_signal_max,na.rm=T)
chipmax.minval

```

```
## [1] -80.85739
```

```
chipmean.log = log(-chipmean.minval + 1 + gr.promoters$chip_signal_mean,base=2)
chipmax.log = log(-chipmax.minval + 1 + gr.promoters$chip_signal_max,base=2)
```

```
gr.promoters$log_chip_signal_mean = chipmean.log
gr.promoters$log_chip_signal_max = chipmax.log
head(gr.promoters)
```

```
## GRanges object with 6 ranges and 7 metadata columns:
```

```
##      seqnames      ranges strand |      len  wbps_gene_id  gene_name
##      <Rle>      <IRanges> <Rle> | <integer>   <character> <character>
## [1]   chrI 26582-27781      - |    1200 WBGene00022278   rcor-1
## [2]   chrI 32951-34150      - |    1200 WBGene00022279   sesn-1
## [3]   chrI 42733-43932      + |    1200 WBGene00022275   txt-7
## [4]   chrI 46461-47660      + |    1200 WBGene00044345  Y48G1C.12
## [5]   chrI 48921-50120      + |    1200 WBGene00021677   pgs-1
## [6]   chrI 63867-65066      - |    1200 WBGene00021678   Y48G1C.5
##      chip_signal_mean chip_signal_max log_chip_signal_mean log_chip_signal_max
##      <numeric>      <numeric>      <numeric>      <numeric>
## [1]      116.59365      220.93678          7.76858          8.24219
## [2]       23.56896       38.75358          6.96620          6.91422
## [3]        7.16118       18.78316          6.76325          6.65307
## [4]       26.93845       43.20576          7.00456          6.96651
## [5]       11.93393       34.69149          6.82529          6.86479
## [6]        -5.76947        9.25825          6.58041          6.50963
```

```
## -----
```

```
## seqinfo: 7 sequences (1 circular) from ce11 genome
```

```
LOG_PROMOTOR_DF_PATH = sprintf("%s/log_filtered.promoters.minus%d_plus%d.df", OUTPUT_03, UPSTREAM, DOWNSTREAM)
write.table(as.data.frame(gr.promoters), file = LOG_PROMOTOR_DF_PATH,quote=F, row.names=F,sep="\t")
```

```
laps = findOverlaps(gr.promoters,gr.IDR, ignore.strand=T,minoverlap = 100)
length(laps)
```

```
## [1] 1358
```

```
head(laps)
```

```
## Hits object with 6 hits and 0 metadata columns:
```

```
##      queryHits subjectHits
##      <integer>   <integer>
## [1]          1           4
## [2]         29           7
## [3]         31           8
## [4]         32           9
## [5]         38          14
## [6]         42          16
```

```
## -----
```

```
## queryLength: 13246 / subjectLength: 4098
```

```
gr.promoters$IDR_mean = NaN
gr.promoters$IDR_max = NaN
gr.promoters$IDR_value = NaN
gr.promoters$nlogq = NaN
gr.promoters[from(laps)]$IDR_max = gr.IDR[to(laps)]$signal.max
gr.promoters[from(laps)]$IDR_mean = gr.IDR[to(laps)]$signal.mean
gr.promoters[from(laps)]$IDR_value = gr.IDR[to(laps)]$intensity
```

```
gr.promoters[from(laps)]$nlogq = gr.IDR[to(laps)]$nlogq
print("Number of promoters overlapping an IDR peak:")
```

```
## [1] "Number of promoters overlapping an IDR peak:"
```

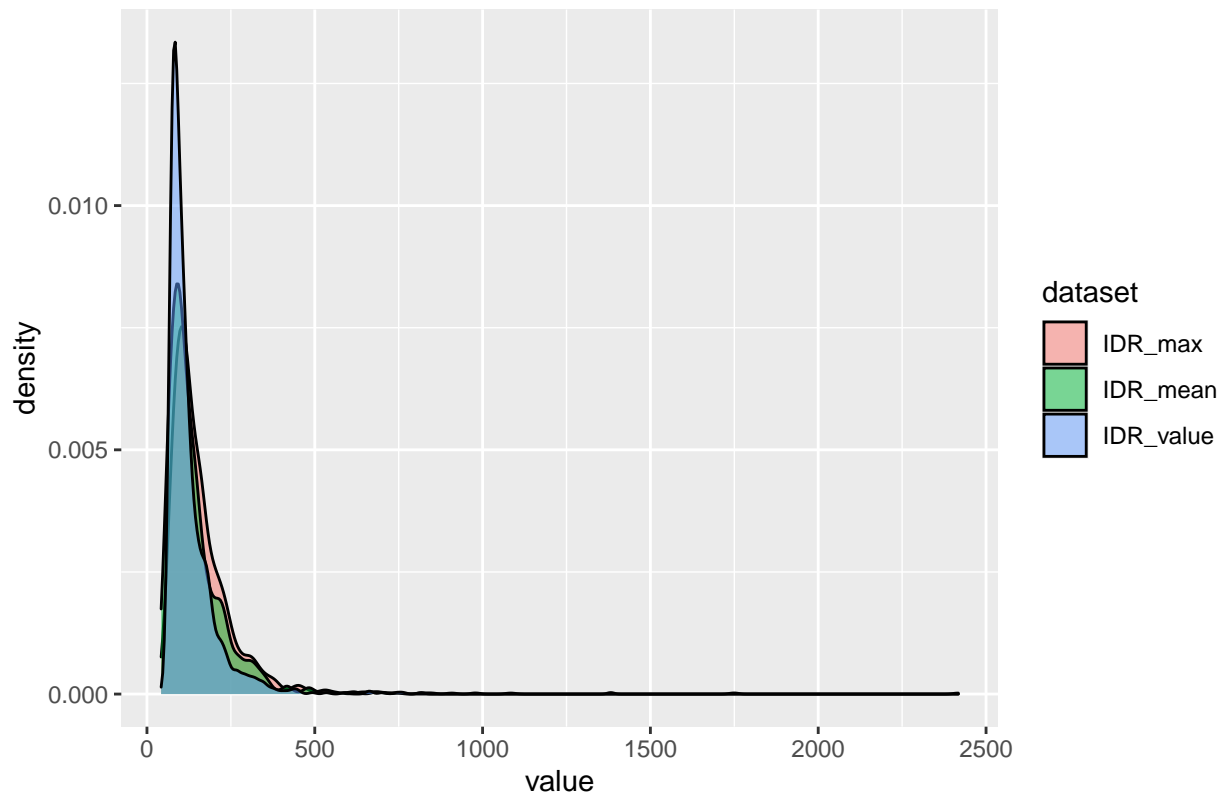
```
sum(!is.nan(gr.promoters$IDR_max))
```

```
## [1] 1275
```

```
idr.nonlog = gather(as.data.frame(gr.promoters)[,c('IDR_value', 'IDR_mean', 'IDR_max')], key="dataset")
ggplot(idr.nonlog, aes(x=value, fill=dataset)) + geom_density(alpha=.5) + labs(title="Distributions of
```

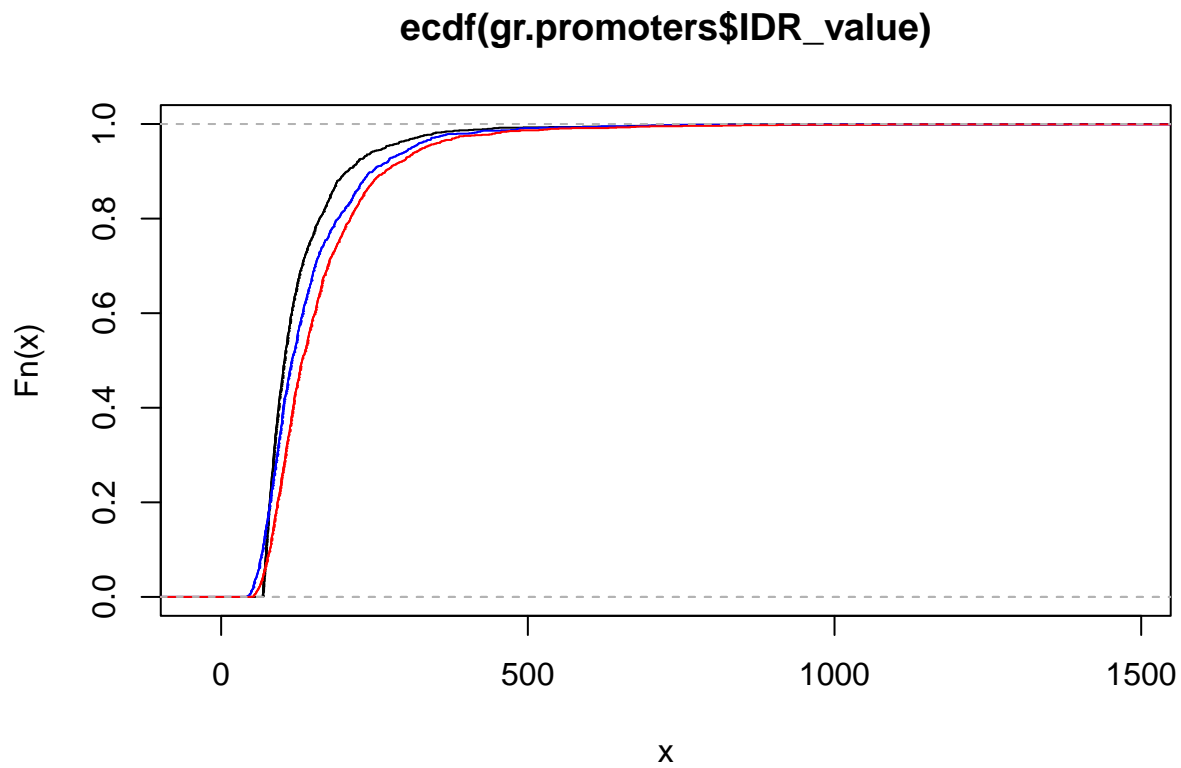
```
## Warning: Removed 35913 rows containing non-finite values (stat_density).
```

### Distributions of log10 transformed IDR NON-Log transformed values



```
idr.val.ecdf = ecdf(gr.promoters$IDR_value)
idr.mean.ecdf = ecdf(gr.promoters$IDR_mean)
idr.max.ecdf = ecdf(gr.promoters$IDR_max)
```

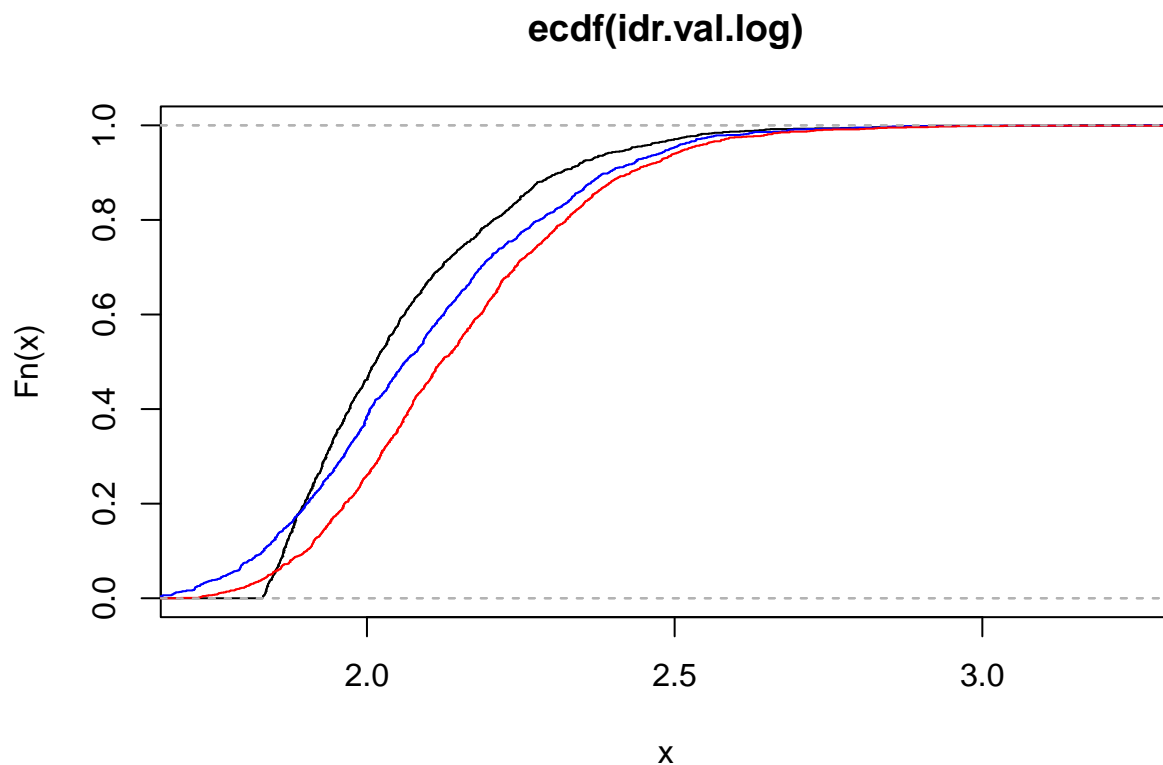
```
plot(idr.val.ecdf)
lines(idr.mean.ecdf, col="blue")
lines(idr.max.ecdf, col="red")
```



```
# the data currently have all positive values, so no adjustment made for log
idr.val.log = log10(gr.promoters$IDR_value)
idr.mean.log = log10(gr.promoters$IDR_mean)
idr.max.log = log10(gr.promoters$IDR_max)

idr.val.log.ecdf = ecdf(idr.val.log)
idr.mean.log.ecdf = ecdf(idr.mean.log)
idr.max.log.ecdf = ecdf(idr.max.log)

plot(idr.val.log.ecdf)
lines(idr.mean.log.ecdf,col="blue")
lines(idr.max.log.ecdf,col="red")
```



```
log.vals = data.frame(idr.mean = idr.mean.log, idr.val = idr.val.log, idr.max = idr.max.log)
```

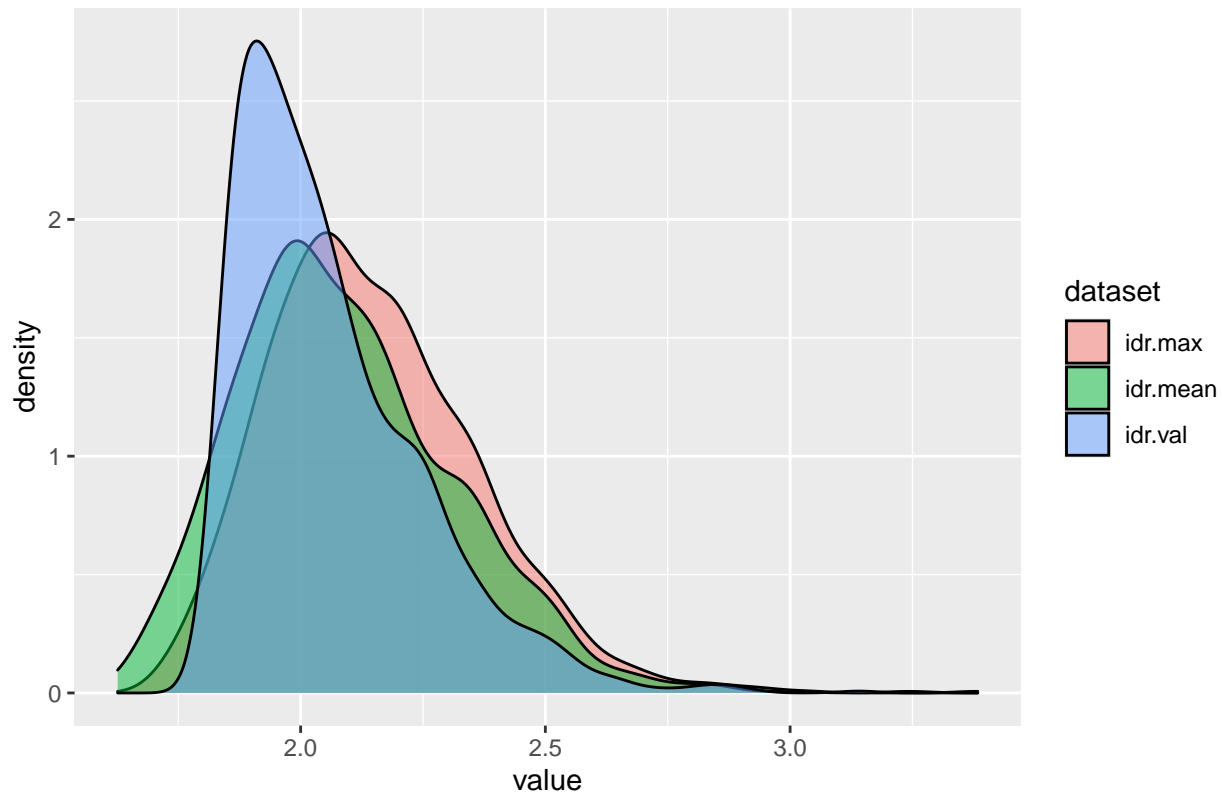
```
long.log.vals = gather(log.vals, key="dataset")
head(long.log.vals)
```

```
##   dataset   value
## 1 idr.mean 2.288332
## 2 idr.mean      NaN
## 3 idr.mean      NaN
## 4 idr.mean      NaN
## 5 idr.mean      NaN
## 6 idr.mean      NaN
```

```
ggplot(long.log.vals, aes(x=value, fill=dataset)) + geom_density(alpha=.5) + labs(title="Distributions")
```

```
## Warning: Removed 35913 rows containing non-finite values (stat_density).
```

## Distributions of log10 transformed IDR values



```
gr.promoters$IDR_logTEN_max = idr.max.log
gr.promoters$IDR_logTEN_mean = idr.mean.log
gr.promoters$IDR_logTEN_value = idr.val.log
sum(idr.mean.log > 2.5, na.rm=T)
```

```
## [1] 59
```

```
datapath = normalizePath('.../Rob/02_embryo_intestine_RNAseq/03_output/DE_Results_GFPplus-vs-GFPmi
x = read.csv(datapath)
rownames(x) <- x$WBGeneID
```

```
# look at the number filtered by DESeq2
# as described by https://bioconductor.org/packages/release/bioc/vignettes/DESeq2/inst/doc/DESeq2.html#
baseMean_is_zero = x$baseMean == 0
pval_na = is.na(x$pvalue)
padj_na = is.na(x$padj)
# case one
sum(baseMean_is_zero & pval_na & padj_na)
```

```
## [1] 0
```

```
# case two
sum(!baseMean_is_zero & pval_na & padj_na)
```

```
## [1] 52
```

```
# case three
sum(!pval_na & padj_na)
```

```
## [1] 3088
```

```
head(x)
```

```
##
##          WBGeneID   baseMean log2FoldChange    lfcSE      pvalue
## WBGene00021406 WBGene00021406 114.300065      0.8215012 0.5991126 9.785429e-02
## WBGene00021407 WBGene00021407  16.099710      0.1920445 0.8458947 6.959493e-01
## WBGene00021408 WBGene00021408  19.527924      8.2813467 2.7319552 2.166056e-07
## WBGene00021405 WBGene00021405   2.279759      0.8617914 1.5230514 4.522192e-02
## WBGene00021409 WBGene00021409   2.405898      0.3925713 1.1231272 9.076469e-02
## WBGene00021404 WBGene00021404  20.940074      9.0546662 2.7383369 3.533428e-09
##
##          padj
## WBGene00021406 1.890764e-01
## WBGene00021407 7.892656e-01
## WBGene00021408 2.186568e-06
## WBGene00021405      NA
## WBGene00021409      NA
## WBGene00021404 4.304637e-08
```

```
#x %>% filter(WBGeneID %in% gr.promoters$wbps_gene_id) -> x.coherent
mccols(gr.promoters) <- mcols(gr.promoters) %>% cbind(x[gr.promoters$wbps_gene_id,2:6]) %>% as.data.frame()
head(gr.promoters)
```

```
## GRanges object with 6 ranges and 19 metadata columns:
##      seqnames      ranges strand |      len  wbps_gene_id  gene_name
##      <Rle>      <IRanges> <Rle> | <integer>  <character> <character>
## [1]   chrI 26582-27781      - |    1200 WBGene00022278    rcor-1
## [2]   chrI 32951-34150      - |    1200 WBGene00022279    sesn-1
## [3]   chrI 42733-43932      + |    1200 WBGene00022275    txt-7
## [4]   chrI 46461-47660      + |    1200 WBGene00044345  Y48G1C.12
## [5]   chrI 48921-50120      + |    1200 WBGene00021677    pgs-1
## [6]   chrI 63867-65066      - |    1200 WBGene00021678    Y48G1C.5
##      chip_signal_mean chip_signal_max log_chip_signal_mean log_chip_signal_max
##      <numeric>      <numeric>      <numeric>      <numeric>
## [1]      116.59365      220.93678          7.76858          8.24219
## [2]      23.56896      38.75358          6.96620          6.91422
## [3]       7.16118      18.78316          6.76325          6.65307
## [4]      26.93845      43.20576          7.00456          6.96651
## [5]      11.93393      34.69149          6.82529          6.86479
## [6]      -5.76947       9.25825          6.58041          6.50963
##      IDR_mean  IDR_max IDR_value IDR_nlogq IDR_logTEN_max IDR_logTEN_mean
##      <numeric> <numeric> <numeric> <numeric>      <numeric>      <numeric>
## [1]    194.237    220.937    199.906    3.57761      2.34427      2.28833
## [2]      NaN      NaN      NaN      NaN      NaN      NaN
## [3]      NaN      NaN      NaN      NaN      NaN      NaN
## [4]      NaN      NaN      NaN      NaN      NaN      NaN
## [5]      NaN      NaN      NaN      NaN      NaN      NaN
## [6]      NaN      NaN      NaN      NaN      NaN      NaN
##      IDR_logTEN_value  baseMean log2FoldChange    lfcSE      pvalue
##      <numeric>      <numeric>      <numeric> <numeric>      <numeric>
## [1]      2.30083    2412.21311     -0.9794165 0.309687 0.000866738
## [2]      NaN    1373.99374     -0.5052768 0.265385 0.047391510
## [3]      NaN     28.90608     -0.5703048 0.613899 0.247771859
## [4]      NaN    1356.79181     -0.4399954 0.319295 0.144923448
## [5]      NaN     548.04398      0.0544551 0.338998 0.867282462
## [6]      NaN      6.18149      0.8158975 1.120646 0.151282869
```



```

##           padj
##      <numeric>
##    [1] 0.00420261
##    [2] 0.10943455
##    [3] 0.37766531
##    [4] 0.25395591
##    [5] 0.91338478
##    [6] 0.26219512
##    -----
##    seqinfo: 7 sequences (1 circular) from ce11 genome

names(gr.promoters) <- gr.promoters$wbps_gene_id

# sort promoters high to low by log2FC
gr.promoters = gr.promoters[order(gr.promoters$log2FoldChange,decreasing=T)]

# divide groups by peak and padj
enriched_intestine = gr.promoters$padj<.05 & !is.na(gr.promoters$padj)
has_peak = !is.na(gr.promoters$IDR_max)
classA = enriched_intestine & has_peak
classB = !enriched_intestine & has_peak
classC = enriched_intestine & !has_peak
classD = !enriched_intestine & !has_peak

gr.promoters$class = "classA"
gr.promoters$class[classB] <- "classB"
gr.promoters$class[classC] <- "classC"
gr.promoters$class[classD] <- "classD"

promoters.hilo = as.data.frame(gr.promoters)

# BED format
write.table(promoters.hilo, file.path(OUTPUT_03, "promoters.hilo.bed"), quote=F, sep="\t", row.names=F,

# Matrix format readable into R
write.table(promoters.hilo, file.path(OUTPUT_03, "promoters.hilo.tsv"), quote=F, sep="\t", row.names=T,

write.table(promoters.hilo[classA,],
            file.path(OUTPUT_03, "promoters.hilo.classA.bed"), quote=F, sep="\t", row.names=F, col.names=
write.table(promoters.hilo[classB,],
            file.path(OUTPUT_03, "promoters.hilo.classB.bed"), quote=F, sep="\t",
row.names=F, col.names=F)
write.table(promoters.hilo[classC,],
            file.path(OUTPUT_03, "promoters.hilo.classC.bed"), quote=F, sep="\t",
row.names=F, col.names=F)
write.table(promoters.hilo[classD,],
            file.path(OUTPUT_03, "promoters.hilo.classD.bed"), quote=F, sep="\t",
row.names=F, col.names=F)

# deeptooling up versus down only, no other filters
promoters.hilo.up = promoters.hilo %>% filter(log2FoldChange > 0)
promoters.hilo.down = promoters.hilo %>% filter(log2FoldChange < 0)

```

```

write.table(promoters.hilo.up,
            file.path(OUTPUT_03, "promoters.hilo.up.bed"),
            quote=F,
            sep="\t",
            row.names=F, col.names=F)

write.table(promoters.hilo.down,
            file.path(OUTPUT_03, "promoters.hilo.down.bed"),
            quote=F,
            sep="\t",
            row.names=F, col.names=F)

```

To produce the deeptools output, execute DEEPTOOLS.bash.

It will compute promoters.hilo.mx and promoters.hilo.pdf.

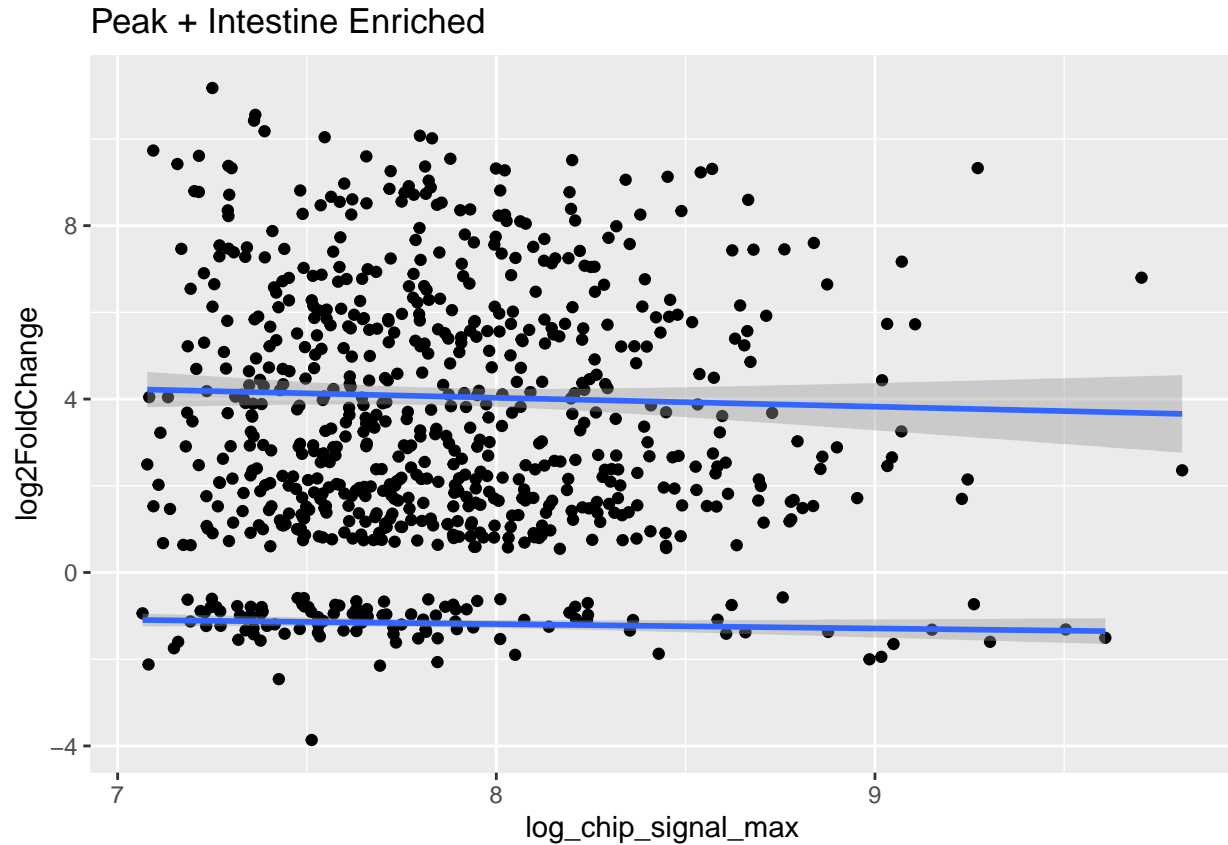
Deeptools PDFs indicate a font called dejavu, if you're tired of replacing it in Illustrator, install it from:  
<https://sourceforge.net/projects/dejavu/>

```

gr.promoters.classA = gr.promoters[classA]

# scatter plot with linear mods on logFC up and down separately
gr.promoters.classA %>% as.data.frame() %>%
  ggplot(
    aes(x=log_chip_signal_max,
        y=log2FoldChange,
        group=log2FoldChange>0)) + geom_point() +
    geom_smooth(method='lm', formula= y~x) +
    ggtitle("Peak + Intestine Enriched")

```



```
classA.up = promoters.hilo %>% as.data.frame() %>% filter(classA & log2FoldChange > 0)
up.table = classA.up[,c('log2FoldChange',
                        'chip_signal_mean',
                        'chip_signal_max',
                        'log_chip_signal_mean',
                        'log_chip_signal_max',
                        'IDR_mean', 'IDR_max', 'IDR_value')]

cor.up.table = cor(up.table)
options(digits=3)
knitr::kable(cor.up.table, caption="Pairwise correlations")
```

Table 1: Pairwise correlations

|                      | log2FoldChange | chip_signal_mean | chip_signal_max | log_chip_signal_mean | log_chip_signal_max | IDR_mean | IDR_max | IDR_value |
|----------------------|----------------|------------------|-----------------|----------------------|---------------------|----------|---------|-----------|
| log2FoldChange       | 1.000          | -0.076           | -0.027          | -0.089               | -0.036              | -0.023   | -0.025  | 0.046     |
| chip_signal_mean     | -0.076         | 1.000            | 0.919           | 0.980                | 0.899               | 0.897    | 0.903   | 0.722     |
| chip_signal_max      | -0.027         | 0.919            | 1.000           | 0.888                | 0.969               | 0.981    | 0.990   | 0.851     |
| log_chip_signal_mean | -0.089         | 0.980            | 0.888           | 1.000                | 0.912               | 0.870    | 0.869   | 0.681     |
| log_chip_signal_max  | -0.036         | 0.899            | 0.969           | 0.912                | 1.000               | 0.955    | 0.957   | 0.796     |
| IDR_mean             | -0.023         | 0.897            | 0.981           | 0.870                | 0.955               | 1.000    | 0.991   | 0.883     |
| IDR_max              | -0.025         | 0.903            | 0.990           | 0.869                | 0.957               | 0.991    | 1.000   | 0.860     |
| IDR_value            | 0.046          | 0.722            | 0.851           | 0.681                | 0.796               | 0.883    | 0.860   | 1.000     |

```
cor.test(classA.up[, 'log2FoldChange'], classA.up[, 'IDR_mean'])
```

```
##  
## Pearson's product-moment correlation  
##  
## data: classA.up[, "log2FoldChange"] and classA.up[, "IDR_mean"]  
## t = -0.6, df = 638, p-value = 0.6  
## alternative hypothesis: true correlation is not equal to 0  
## 95 percent confidence interval:  
## -0.1004 0.0545  
## sample estimates:  
## cor  
## -0.0231
```

```
cor.test(classA.up[, 'log2FoldChange'], classA.up[, 'log_chip_signal_mean'])
```

```
##  
## Pearson's product-moment correlation  
##  
## data: classA.up[, "log2FoldChange"] and classA.up[, "log_chip_signal_mean"]  
## t = -2, df = 638, p-value = 0.02  
## alternative hypothesis: true correlation is not equal to 0  
## 95 percent confidence interval:  
## -0.1654 -0.0117  
## sample estimates:  
## cor  
## -0.0891
```