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 ce11 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(overlappingPromoterRo
```

```
## [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.minus10
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_subtracte
```

```
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

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.

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, DOWNS
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
```

```
datapath = normalizePath('../../../Rob/02_embryo_intestine_RNAseq/03_output/DE_Results_GFPplus-vs-GFPmin
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$wbps_gene_id) -> x.coherent
mcols(gr.promoters) <- mcols(gr.promoters) %>% cbind(x[gr.promoters$wbps_gene_id,2:6])  %>% as.data.fra
head(gr.promoters)
```

```
## GRanges object with 6 ranges and 16 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   baseMean log2FoldChange
##       <numeric> <numeric> <numeric> <numeric>  <numeric>      <numeric>
##   [1]   194.237   220.937   199.906   3.57761 2412.21311     -0.9794165
##   [2]       NaN       NaN       NaN       NaN 1373.99374     -0.5052768
##   [3]       NaN       NaN       NaN       NaN   28.90608     -0.5703048
##   [4]       NaN       NaN       NaN       NaN 1356.79181     -0.4399954
##   [5]       NaN       NaN       NaN       NaN  548.04398      0.0544551
##   [6]       NaN       NaN       NaN       NaN    6.18149      0.8158975
##           lfcSE      pvalue       padj
##       <numeric>   <numeric>  <numeric>
##   [1]  0.309687 0.000866738 0.00420261
##   [2]  0.265385 0.047391510 0.10943455
##   [3]  0.613899 0.247771859 0.37766531
##   [4]  0.319295 0.144923448 0.25395591
##   [5]  0.338998 0.867282462 0.91338478
##   [6]  1.120646 0.151282869 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.nan(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.name
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)
```

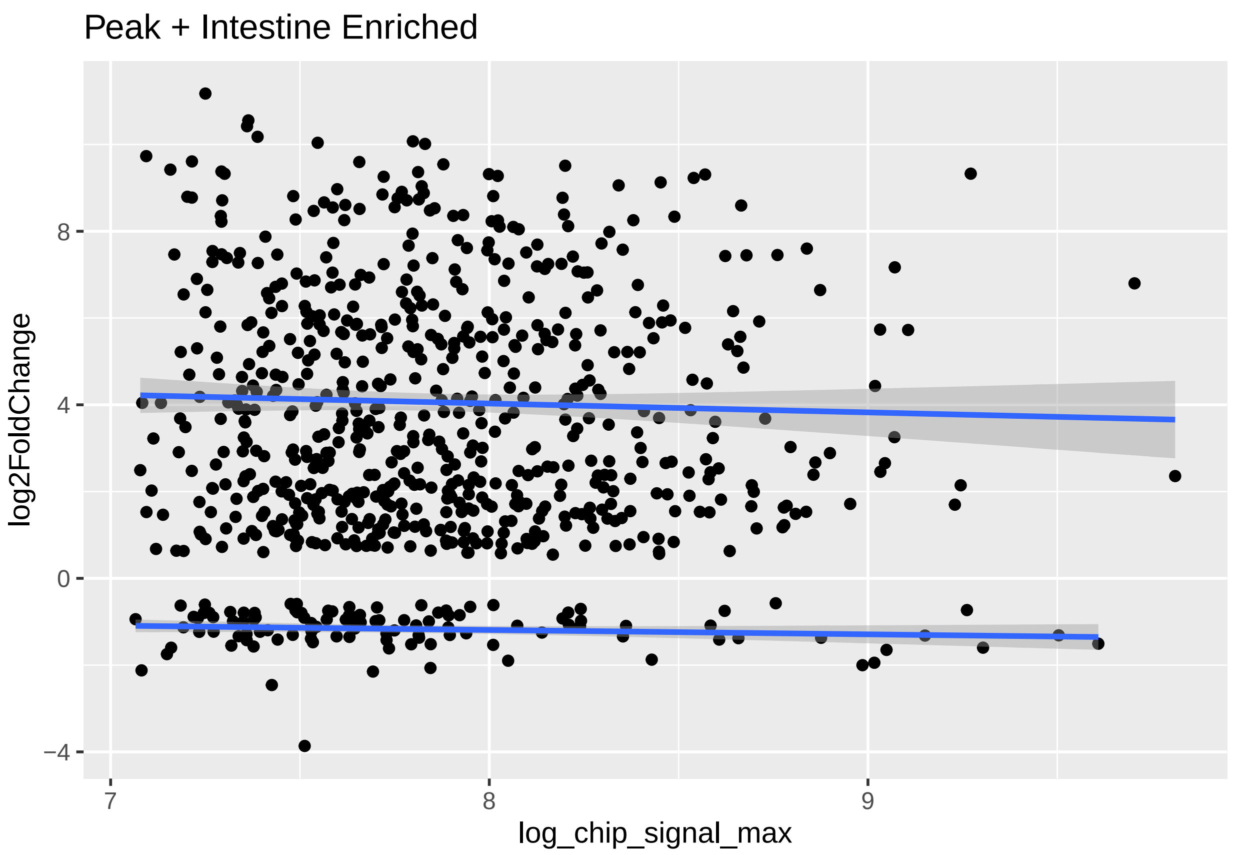
To produce the deeptools output, execute DEEPTOOLS.bash.

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)
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	1.0000000	-0.0760979	-0.0272986	-0.0890844	-0.0355203	-0.0230808	-0.0253050	0.0457812
chip_signal_mean	-0.0760979	1.0000000	0.9194545	0.9800102	0.8994858	0.8974257	0.9029277	0.7222291
chip_signal_max	-0.0272986	0.9194545	1.0000000	0.8881889	0.9687861	0.9807173	0.9902824	0.8514666
log_chip_signal_mean	-0.0890844	0.9800102	0.8881889	1.0000000	0.9117432	0.8698112	0.8693059	0.6814238
log_chip_signal_max	-0.0355203	0.8994858	0.9687861	0.9117432	1.0000000	0.9545215	0.9571380	0.7964375
IDR_mean	-0.0230808	0.8974257	0.9807173	0.8698112	0.9545215	1.0000000	0.9906743	0.8833812

	log2FoldChange	chip_signal_mean	chip_signal_max	log_chip_signal_mean	log_chip_signal_max	IDR_mean	IDR_max	IDR_value
IDR_max	-0.0253050	0.9029277	0.9902824	0.8693059	0.9571380	0.9906743	1.0000000	0.8596847
IDR_value	0.0457812	0.7222291	0.8514666	0.6814238	0.7964375	0.8833812	0.8596847	1.0000000

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

```
##
##  Pearson's product-moment correlation
##
## data:  classA.up[, "log2FoldChange"] and classA.up[, "IDR_mean"]
## t = -0.58315, df = 638, p-value = 0.56
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.10040210  0.05451757
## sample estimates:
##         cor
## -0.02308082
```

```
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 = -2.2591, df = 638, p-value = 0.02421
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.16544303 -0.01166405
## sample estimates:
##        cor
## -0.0890844
```