PO.EN Demonstration

Introduction

This document describes a complete walk through the usage of the package 'PO.EN' with an application to predicting regulatory effects of genetic variants at a GWAS locus. These are the main steps:

- Pre-process results from massively parallel reporter assays (MPRAs) to generate the presence-only responses.
- Extract DeepSEA epigenetic features.
- Fit the PO.EN model on the training data.
- Make predictions based on the fitted model for a position of interest (at sequence level).

Pre-process MPRAs dataset

The dataset used to generate the presence-only response for this illustration was generated using the SuRE assay [van Arensbergen et al., 2019], an MPRA able to systematically screen millions of SNPs for potential regulatory effects.

```r
setwd('../wd') # Set up the working directory
# Load a subset of SuRE dataset
sure<-read.csv('sure_example.csv',header=T)
head(sure[,c(1:3,8,12,14:15)])
```

```
##     chr      SNP_ID SNPabspos k562.wilcox.p.value hepg2.wilcox.p.value ref alt
## 1  chr5  rs10056572   1559341        2.030421e-33         1.218414e-30   T   C
## 2 chr18    rs991512  68919785        7.220797e-33         7.516356e-32   C   G
## 3 chr14   rs7156067  89393045        2.905157e-31         2.073221e-35   G   A
## 4  chr4   rs1605769  59436721        2.977554e-31         2.481365e-36   T   C
## 5 chr14   rs1204985  72218104        1.816043e-29         5.527318e-04   A   G
## 6 chr19 rs117788163  46743620        2.660431e-29         1.715368e-23   C   G
```

The size of the complete SuRE dataset is too large to handle for this documentation. Hence, the SuRE data in the code above is only the first 100 observations, just as an illustration. The actual training datasets, which will be shown in later sections, are still generated based on the complete SuRE dataset.

We use the values in 'sure$k562.wilcox.p.value' and 'sure$hepg2.wilcox.p.value' to generate the presence-only responses for the models related to the cell lines K562 and HepG2. Let $\mathbf{p}$ denote the vectors of the P values in the SuRE dataset (either K562 or HepG2), and $\mathbf{y}$ denote the presence-only response vector. For any $i \in \{1, \ldots\}$, to generate $y_i \in \mathbf{y}$ for K562 or HepG2, we set the threshold of P values for identifying a SNP as functional to be 10^{-5}, i.e.,

$$y_i = \begin{cases} 1 & \text{if } p_i \leq 10^{-5} \\ 0 & \text{if } p_i > 10^{-5} \end{cases} \tag{1}$$

GeneHancer

Before actually generating the presence-only response vector, we first use the GeneHancer database to identify variants residing in putative enhancer elements for training.

```
genehancer<-read.csv('genehancer_example.csv',header=T)
head(genehancer)
```

```
##     chr     start       end                      id score
## 1 chr17   2074111   2087474    chr17:2074111-2087474  3.11
## 2 chr15  97128054  97130630  chr15:97128054-97130630  3.03
## 3  chr8  49491647  49497018   chr8:49491647-49497018  3.02
## 4  chr5  77146826  77149236   chr5:77146826-77149236  2.95
## 5 chr20  21214790  21217232  chr20:21214790-21217232  2.93
## 6  chr1 213498112 213501390 chr1:213498112-213501390  2.92
```

For the training dataset, we select those SNPs in the SuRE dataset that fall into enhancer elements that have the corresponding 'genehancer$score' larger than 1 in the GeneHancer dataset. By doing so, roughly 700k SNPs are selected for each cell line. Among these SNPs, we include in the training datasets for K562 and HepG2 all the SNPs with the corresponding P values smaller than 10^{-5} (positive examples), and we select a set of background variants with 1:20 ratio of presence:background in the training dataset.

The following code shows an example of mapping the SNPs on the SuRE assay to putative enhancer elements in GeneHancer, which relies on the R package 'rtracklayer'.

```
#create GRanges object for SuRE
genehancer.gr<-GRanges(Rle(genehancer$chr),IRanges(start=genehancer$start,end=genehancer$end))
#create GRanges object for GeneHancer
sure.gr<-GRanges(Rle(sure$chr),IRanges(start=sure$SNPabspos,end=sure$SNPabspos))
#Mapping the two datasets
mapping<-as.data.frame(findOverlaps(sure.gr,genehancer.gr,type='within'))
# Identify the SuRE SNPs with GeneHancer's scores larger than 1
sure_screened<-sure[mapping[which(genehancer[mapping[,2],]$score>1),1],]
imba<-20 # imbalance ratio
# Use HepG2 tissue as an example
presence.index<-which(sure_screened$hepg2.wilcox.p.value<=1e-5)
absence.index<-sample((1:nrow(sure_screened))[-presence.index],length(presence.index)*imba)
sure_final<-sure_screened[c(presence.index,absence.index),]
```

The binary indicators of the P values given the threshold 10^{-5} in 'sure_final' is the presence-only responses that we will be using for training the models.

Extracting DeepSEA features

The SuRE dataset includes the information on SNPs' chromosomes, ID, positions, reference and alternative alleles, which are used to extract the DeepSEA epigenetic features [J and OG, 2015]. The following codes show how to generate the file that the DeepSEA requires.

```
ex.vcf<-data.frame(sure_final[,c(1,3)],1:nrow(sure_final),sure_final[,14:15])
write.table(ex.vcf,file=paste0('vcf_file.vcf'),sep='\t',quote=F,row.names = F,col.names = F)
```

After retrieving the DeepSEA features, the following code shows how to create the final training dataset.

```
deepsea<-read.table(paste0('infile.vcf.out.evalue'),sep=",",header = T)
train_data<-cbind(sure_final[deepsea$name,],deepsea[,-c(1:6)])
```

```
train_data<-read.csv( 'HepG2_SuRE.csv',header = T)
# only show the first six columns of the training dataset
head(train_data[,1:6])
```

```
##     chr ref.alt       pos HepG2.p.value X8988T.DNase.None AoSMC.DNase.None
## 1 chr14     A-G  31735276  2.705026e-08         0.2576000         0.811300
## 2 chr10     G-T  81059335  3.789795e-06        0.0082311          0.019694
## 3 chr17     A-G  67607006  5.014499e-21        0.8351400          0.399750
## 4 chr13     A-C  26759189  1.424794e-06        0.0673890          0.216620
## 5  chr6     A-G 170352443  8.896305e-07        0.3107700          0.946520
## 6 chr18     T-C  40286816  3.085195e-06        0.0160710          0.100700
```

Fitting the models

First, loading the 'PO.EN' package.

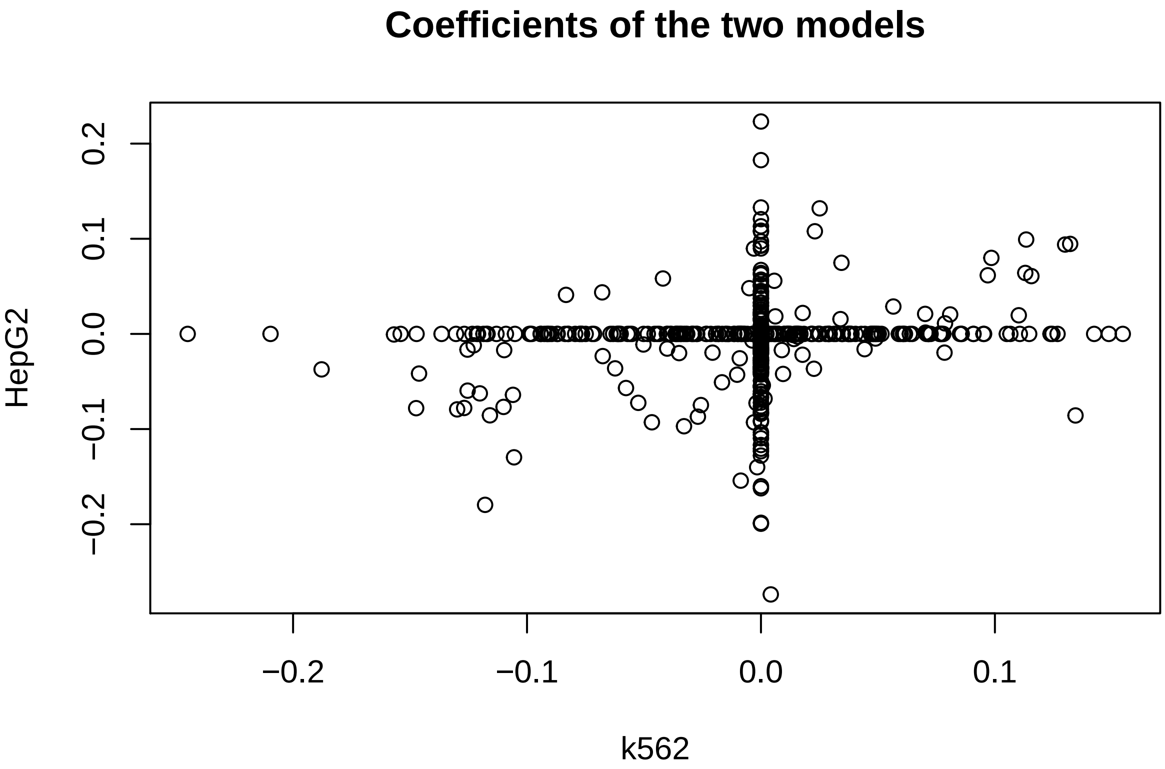
```
install.packages('PO.EN')
library("PO.EN")
```

Running the cross-validation function first, acquire the values of the tuned λ and π, then fit the model with the pair of tuned values.

```
# Presence-only response vector
y<-as.numeric((train_data$HepG2.p.value<=1e-5))
# DeepSEA features
x<-train_data[,(ncol(train_data)-918):ncol(train_data)]
x<-scale(x)
cv<-cv.PO.EN(x,y,nfolds = 10,input.pi=seq(0.05,0.45,length.out=10))
PO.EN.beta<-PO.EN(x,y,lambda = cv$lambda.min,true.prob=cv$pi,beta_start=rep(0,ncol(x)+1))
```

Typically, fitting the model ('PO.EN' function) is fairly fast, but the cross-validation part is more time consuming. The following code shows the estimated coefficients vectors in the K562 and HepG2 models.

```
k562<-read.csv('k562_coefs.csv')
hepg2<-read.csv('hepg2_coefs.csv')
plot(k562[-1,1],hepg2[-1,1],ylab='HepG2',xlab='k562',main='Coefficients of the two models')
```



As we can observe, many coefficients have been shrunk to zero, and the two models share some non-zero coefficients.

Making predictions

Suppose that we are interested in assessing the regulatory effect of a specific variant or set of variants in a region, such as a GWAS locus. The example below is for the SORT1 GWAS locus, i.e. a 100Kb region centered around 'rs12740374'. We select all the SNPs included in the SuRE dataset (although we could also make predictions at any position in this region), extract the corresponding DeepSEA features, build the design matrix, and make predictions for these SNPs.

```
# Generate GRanges object for the functional SNPs with 100000 bp
snps.gr<-GRanges(Rle('chr1'),IRanges(start=109817590-50000,end = 109817590+50000))
# Map SuRE to this neighborhood range
mapping<-as.data.frame(findOverlaps(sure.gr,snps.gr,type = 'within'))
# Create vcf file to extract DeepSEA
sure_snps<-sure[mapping[,1],]
ex.vcf<-data.frame(sure_snps[,c(1,3)],1:nrow(sure_snps),sure_snps[,14:15])
write.table(ex.vcf,file=paste0('vcf_snps.vcf'),sep='\t',quote=F,row.names = F,col.names = F)
```

After extracting DeepSEA features, we first scale the new design matrix, then make predictions.

```
deepsea_sort1<-read.table('sort1_infile.vcf.out.evalue',sep=',',header=T)
x_test<-scale(deepsea_sort1[,(ncol(deepsea_sort1)-918):ncol(deepsea_sort1)])
#Using the prediction function in PO.EN package
k562_predictions<-PO.EN.predict(x_test,k562[,1])
hepg2_predictions<-PO.EN.predict(x_test,hepg2[,1])
#Merge the predictions and the original SuRE data,
sort1<-cbind(sure_snps[deepsea_sort1$name,],k562_predictions,hepg2_predictions)
#Save the predictions first
write.csv(sort1,file='SORT1.csv',row.names = F)
```

Making plots

We use the package 'ggplot' to plot the results. The plots will be separated into two parts based on the two cell lines.

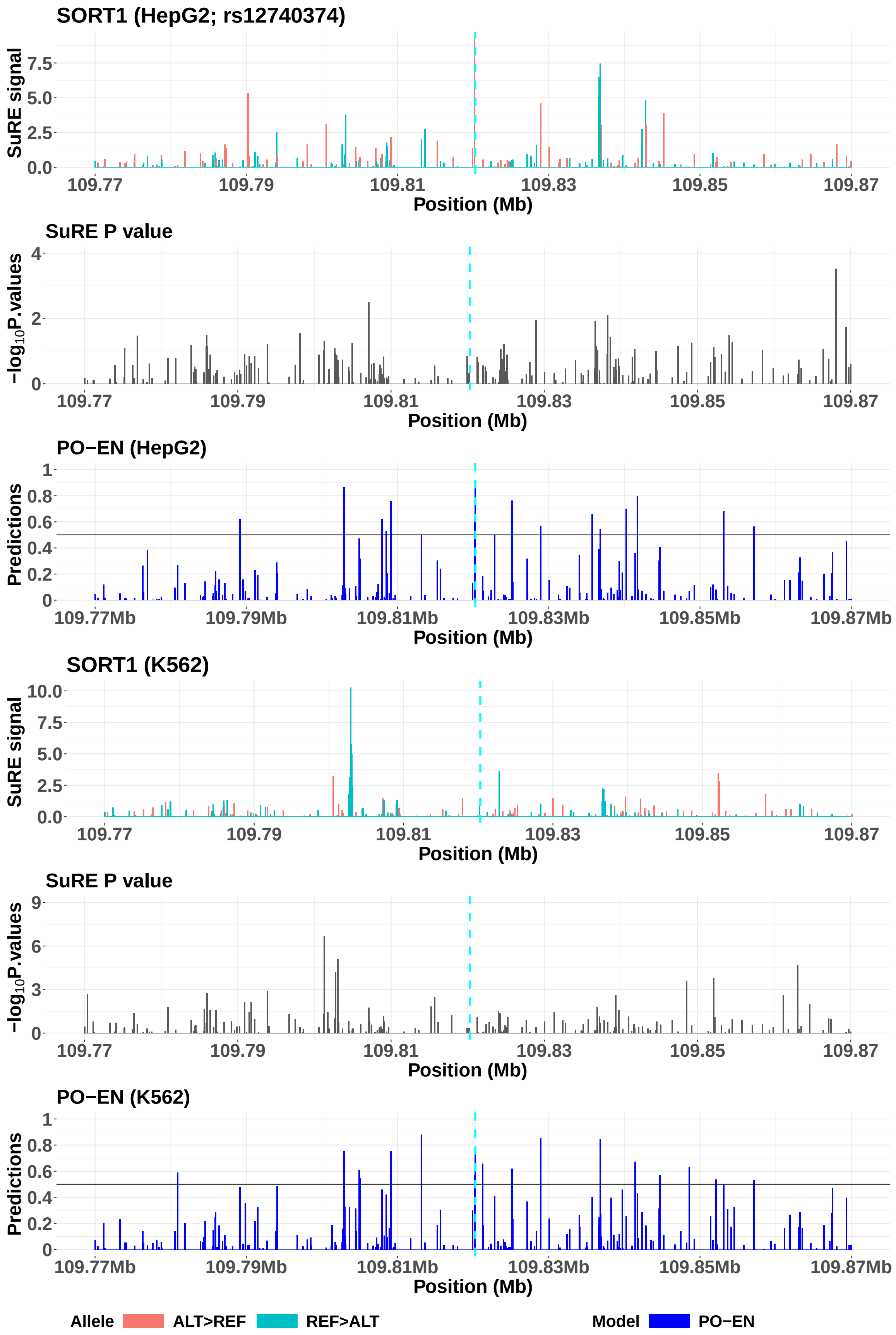
The following codes show how to make plot

```
library(ggplot2)
library(gridExtra)
library(grid)
library(gtable)
#read the predictions and original dataset
data.list<-list()
data.list[[1]]<-read.csv('SORT1.csv',header = T)

cell.type=c('HepG2','K562')
u.gene.name='SORT1'
SNP_ID='rs12740374'
title<-c(paste0(u.gene.name,' (' ,cell.type[1],'; ',
                paste0(as.character(SNP_ID),collapse = ',') ,')'),
         paste0(u.gene.name,' (' ,cell.type[2],')'))
#Make plot
snp.index.list<-list()
plot.data.list<-list()
plot.list<-list()
```

```r
break.list<-list()

  for(h in 1:2){
    # Select columns according to the tissue
  if(cell.type[h]=='HepG2'){
    plot.data<-data.list[[1]][,c(1:5,10:12,16)]
    colnames(plot.data)[9]<-c('PO-EN (HepG2)')
  }else{
    plot.data<-data.list[[1]][,c(1:8,18)]
    colnames(plot.data)[9]<-c('PO-EN (K562)')
  }
  #Sort the position
plot.data<-plot.data[order(plot.data[,3]),]
  #Transform into real sequence
p.plot.data<-data.frame(as.data.frame(
  matrix(NA,nrow=plot.data$SNPabspos[nrow(plot.data)]-plot.data$SNPabspos[1]+1,
         ncol=ncol(plot.data))))
colnames(p.plot.data)=colnames(plot.data)
p.plot.data$chr=plot.data$chr[1]
p.plot.data$SNPabspos=plot.data$SNPabspos[1]:plot.data$SNPabspos[nrow(plot.data)]
p.plot.data[match(plot.data$SNPabspos,p.plot.data$SNPabspos),]=plot.data
p.plot.data$SNP_ID[match(plot.data$SNPabspos,p.plot.data$SNPabspos)]<-
  as.character(plot.data$SNP_ID)
p.plot.data[is.na(p.plot.data)]=0
snp.index.list[[1]]<-match(SNP_ID,p.plot.data$SNP_ID)
print(snp.index.list[[1]])
print(dim(p.plot.data))

plot.data.list[[h]]<-list()
#SuRE signal
plot.data.list[[h]][[1]]<-data.frame(
signal=abs(p.plot.data[,6]-p.plot.data[,7]),
pos=1:nrow(p.plot.data),
Allele=ifelse(p.plot.data[,6]-p.plot.data[,7]>=0,'REF>ALT','ALT>REF'),
snppos=round(p.plot.data$SNPabspos/1000000,2)
)
#SuRE P values
plot.data.list[[h]][[2]]<-data.frame(
signal=abs(log10(p.plot.data[,8])),
pos=1:nrow(p.plot.data),
snppos=round(p.plot.data$SNPabspos/1000000,2)
)
#PO.EN predictions
plot.data.list[[h]][[3]]<-data.frame(
signal=c(p.plot.data[,9]),
pos=rep(1:nrow(p.plot.data),1),
Model=rep(c('PO-EN'),each=nrow(p.plot.data)),
snppos=round(p.plot.data$SNPabspos/1000000,2)
)

break.list[[h]]<-floor(seq(1,nrow(plot.data.list[[h]][[1]]),length.out = 6))

plot.list[[h]]<-list()
```

```
head(plot.data.list[[h]][[1]])
plot.list[[h]][[1]]<-ggplot(data=plot.data.list[[h]][[1]],aes(x=pos, y=signal,fill=Allele))
+ggtitle(title[h])+ylim(0,max(plot.data.list[[h]][[1]]$signal))+
  scale_x_continuous(breaks=(plot.data.list[[h]][[1]]$pos)[break.list[[h]]],
                     labels=paste0(plot.data.list[[h]][[1]]$snppos[break.list[[h]]] ))+
  geom_bar(stat="identity",width=200)+xlab('Position (Mb)')+ylab('SuRE signal')+theme_bw()+
  theme(legend.key.width = unit(1.8,"cm"), legend.position="bottom",
        axis.text.x=element_text(size=15,face="bold"),axis.text.y =element_text(size=12),
        axis.title=element_text(size=14),  plot.title = element_text(size=18,face="bold"),
        legend.title = element_text( size=20,face="bold"),
        legend.text = element_text( size=20,  face="bold"))
for(s in 1:length(snp.index.list[[1]])){
  plot.list[[h]][[1]] <-plot.list[[h]][[1]]+
    geom_vline(xintercept =snp.index.list[[1]][s],col='cyan1',linetype='dashed',size=1.2)
}

plot.data.list[[h]][[2]]$signal[is.infinite(plot.data.list[[h]][[2]]$signal)]<-0
p2.max<-max(plot.data.list[[h]][[2]]$signal)
break.index<-floor(floor(p2.max)/5)
if(break.index==0){
  break.index=1
}
p2.break<-(0:floor(p2.max))[c(TRUE, rep(FALSE,abs(break.index)))]
if(length(p2.break)==2){
  p2.break<-c(p2.break,p2.break[2]*2)
}
if(length((p2.break))==1){
  p2.break<-c(p2.break,1,round(p2.max))
}

plot.list[[h]][[2]]<-ggplot(data=plot.data.list[[h]][[2]], aes(x=pos, y=signal))+
  ggtitle('SuRE P value') +
  scale_y_continuous(breaks=p2.break,limits = c(0,p2.break[length(p2.break)]),
                     labels=as.character(p2.break))+
  geom_bar(stat="identity",width=200)+
  xlab('Position (Mb)')+
  ylab(expression(paste(bold('-log')[10],bold('P.values'))))+
  theme_bw()+
  scale_x_continuous(breaks=(plot.data.list[[h]][[1]]$pos)[break.list[[h]]],
                     labels=paste0(plot.data.list[[h]][[2]]$snppos[break.list[[h]]] ))+
  theme(axis.text.x=element_text(size=15,face="bold"),
        axis.text.y =element_text(size=12),
        axis.title=element_text(size=14),
        plot.title = element_text(size=18,face="bold"),
        legend.title = element_text( size=20,face="bold"),
        legend.text = element_text( size=20,  face="bold"))
for(s in 1:length(snp.index.list[[1]])){
  plot.list[[h]][[2]]<-plot.list[[h]][[2]]+
    geom_vline(xintercept =snp.index.list[[1]][s],col='cyan1',linetype='dashed',size=1.2)
}

  plot.list[[h]][[3]]<-ggplot(data=plot.data.list[[h]][[3]],
                              aes(x=pos, y=signal,fill=Model))+
    theme_bw()+
    scale_fill_manual(values=c('blue1'))+
    scale_x_continuous(breaks=(plot.data.list[[h]][[3]]$pos)[break.list[[h]]],
                       labels=paste0(plot.data.list[[h]][[3]]$snppos[break.list[[h]]],'Mb'))+
    geom_bar(stat="identity", position=position_dodge(),width=200)+
    ggtitle(colnames(plot.data)[j+6])+xlab('Position (Mb)')+
    geom_hline(yintercept=0.5, linetype="solid", color = "grey3")+
    ylab('Predictions')+
    scale_y_continuous(breaks=c(0,0.2,0.4,0.6,0.8,1),
                       limits = c(0,1), labels=c('0','0.2','0.4','0.6','0.8','1' ))+
    theme( legend.key.width = unit(1.8,"cm"),
           legend.position="bottom",
           axis.text.x=element_text(size=15,face="bold"),
           axis.text.y =element_text(size=12) ,
           axis.title=element_text(size=14),
           plot.title = element_text(size=18,face="bold"),
           legend.title = element_text( size=20,face="bold"),
           legend.text = element_text( size=20,  face="bold"))

  for(s in 1:length(snp.index.list[[1]])){
    plot.list[[h]][[3]]<-plot.list[[h]][[3]]+
      geom_vline(xintercept =snp.index.list[[1]][s],col='cyan1',linetype='dashed',size=1.2)
  }


}
  legend = gtable_filter(ggplot_gtable(ggplot_build(plot.list[[1]][[1]])), "guide-box")
  legend2= gtable_filter(ggplot_gtable(ggplot_build(plot.list[[1]][[3]])), "guide-box")
  ml<-grid.arrange(arrangeGrob(
    plot.list[[1]][[1]] +
    theme(panel.border = element_blank(),
    legend.position="none",axis.text.x=element_text(size=22,face="bold"),
    axis.text.y =element_text(size=22,face='bold'),
    axis.title=element_text(size=23,face="bold"),
    plot.title = element_text(size=26,face="bold")),
    plot.list[[1]][[2]] + theme(panel.border = element_blank(),
      legend.position="none",
      axis.text.x=element_text(size=22,face="bold"),
      axis.text.y =element_text(size=22,face='bold'),
      axis.title=element_text(size=23,face="bold"),
      plot.title = element_text(size=24,face="bold")),
  plot.list[[1]][[3]] + theme(panel.border = element_blank(),
       legend.position="none",
       axis.text.x=element_text(size=22,face="bold"),
       axis.text.y =element_text(size=22,face='bold'),
       axis.title=element_text(size=23,face="bold"),
       plot.title = element_text(size=24,face="bold")),
  plot.list[[2]][[1]] + theme(panel.border = element_blank(),
      legend.position="none",
      axis.text.x=element_text(size=22,face="bold"),
      xis.text.y =element_text(size=22,face='bold'),
      axis.title=element_text(size=23,face="bold"),
      plot.title = element_text(size=26,face="bold")),
  plot.list[[2]][[2]] + theme(panel.border = element_blank(),
      legend.position="none",
      axis.text.x=element_text(size=22,face="bold"),
      axis.text.y =element_text(size=22,face='bold'),
      axis.title=element_text(size=23,face="bold"),
      plot.title = element_text(size=24,face="bold")),
  plot.list[[2]][[3]] + theme(panel.border = element_blank(),
      legend.position="none",
      axis.text.x=element_text(size=22,face="bold"),
      axis.text.y =element_text(size=22,face='bold'),
      axis.title=element_text(size=23,face="bold"),
      plot.title = element_text(size=24,face="bold")),
  nrow = 6,ncol=1),arrangeGrob(legend,legend2,nrow=1,ncol=2),ncol=1,nrow=2,heights=c(10,.3))


ggsave(paste0(as.character(u.gene.name[i]),'_normal_distance', '.pdf'),
       ml, width  = unit(30/2, "cm"), height = unit(45/2, "cm"))
```

```
library(knitr)
include_graphics('SORT1_normal_distance.pdf')
```


SORT1 (HepG2; rs12740374)
SuRE signal
7.5
5.0
2.5
0.0
109.77
109.79
109.81
109.83
109.85
109.87
Position (Mb)
SuRE P value
$-\log_{10}$P.values
4
2
0
109.77
109.79
109.81
109.83
109.85
109.87
Position (Mb)
PO−EN (HepG2)
Predictions
1
0.8
0.6
0.4
0.2
0
109.77Mb
109.79Mb
109.81Mb
109.83Mb
109.85Mb
109.87Mb
Position (Mb)
SORT1 (K562)
SuRE signal
10.0
7.5
5.0
2.5
0.0
109.77
109.79
109.81
109.83
109.85
109.87
Position (Mb)
SuRE P value
$-\log_{10}$P.values
9
6
3
0
109.77
109.79
109.81
109.83
109.85
109.87
Position (Mb)
PO−EN (K562)
Predictions
1
0.8
0.6
0.4
0.2
0
109.77Mb
109.79Mb
109.81Mb
109.83Mb
109.85Mb
109.87Mb
Position (Mb)
Allele
ALT>REF
REF>ALT
Model
PO−EN

References

Zhou J and Troyanskaya OG. Predicting effects of noncoding variants with deep learning-based sequence model. *Nature Methods*, 12(10):931–934, 2015.

Joris van Arensbergen, Ludo Pagie, Vincent D FitzPatrick, Marcel de Haas, Marijke P Baltissen, Federico Comoglio, Robin H van der Weide, Hans Teunissen, Urmo Võsa, Lude Franke, et al. High-throughput identification of human snps affecting regulatory element activity. *Nature genetics*, 51(7):1160, 2019.