Analysis of the time-dependent response to Estradiol through ChIP-Seq data and the VULCAN package

Federico M. Giorgi

26 November, 2018

Contents

Introduction

Vulcan (VirtUaL ChIP-Seq Analysis through Networks) is a pipeline that combines ChIP-Seq data and regulatory networks to obtain transcription factors that are likely affected by a specific stimulus. In order to do so, our package combines strategies from different BioConductor packages: *DESeq* for data normalization, *ChIPpeakAnno* and *DiffBind* for annotation and definition of ChIP-Seq genomic peaks, *csaw* to define optimal peak width and *viper* for applying a regulatory network over a differential binding signature. Usage of gene regulatory networks to analyze biological systems has witnessed an exponential increase in the last decade, due to the ease of obtaining genome-wide expression data (Margolin et al., 2005; Giorgi et al., 2013; Castro et al., 2015). Recently, the VIPER approach to interrogate these network models has been proposed to infer transcription factor activity using the expression of a collection of their putative targets, i.e. their regulon (Alvarez et al., 2016). In the VIPER algorithm, gene-level differential expression signatures are obtained for either individual samples (relative to the mean of the dataset) or between groups of samples, and regulons are tested for enrichment. Ideally, TF regulons can be tested on promoter-level differential binding signatures generated from ChIP-Seq experiments, in order to ascertain the global change in promoter occupancy for gene sets. In our study, we propose an extension of the VIPER algorithm to specifically analyze TF occupancy in ChIP-Seq experiments. Our VULCAN algorithm uses ChIP-Seq data obtained for a given TF to provide candidate coregulators of the response to a given stimulus. The analysis is based on identifying differentially bound genes and testing their enrichment in the regulon of potential co-regulatory factors.

Software setup

VULCAN is conveniently provided as an R package available from the Bioconductor repository.

```
#source("http://bioconductor.org/biocLite.R")
#biocLite("vulcan")
library(vulcan)
```

Other packages are required for the execution of the analysis and the visualization of the results. A *results* folder will be also created to store intermediate steps of the analysis.

```
library(gplots) # for heatmap.2
library(org.Hs.eg.db)
library(gridExtra) # for the pathway part
library(GeneNet) # to generate partial correlation networks
if(!file.exists("results")){dir.create("results")}
```

Finally, we will define here some convenience functions, such as those to convert from gene symbols to entrez ids, and viceversa.

```
list_eg2symbol<-as.list(org.Hs.egSYMBOL[mappedkeys(org.Hs.egSYMBOL)])
e2s<-function(ids){
  ids <- as.character(ids)
  outlist <- list_eg2symbol[ids]
  names(outlist) <- ids
  outlist[is.na(outlist)] <- paste("unknown.", ids[is.na(outlist)], sep = "")
  outlist <- gsub("unknown.unknown.", "", outlist)
  return(outlist)
}
list_symbol2eg <- as.character(org.Hs.egSYMBOL2EG[mappedkeys(org.Hs.egSYMBOL2EG)])
s2e<-function(ids){
  ids <- as.character(ids)
  outlist <- list_symbol2eg[ids]
  names(outlist) <- ids
```

```
  outlist[is.na(outlist)] <- paste("unknown.", ids[is.na(outlist)], sep = "")
  outlist <- gsub("unknown.unknown.", "", outlist)
  return(outlist)
}
```

Import ChIP-Seq data using Vulcan

ChIP-Seq data was generated using a cell line model of ER+ breast cancer, MCF7, at 0', 45' and 90' after stimulation with estradiol. The binding profile of ER at each timepoint was then compared between time points using differential binding analysis. From the temporal comparison ER binding we established four classes of binding pattern: early coactivators, early corepressors, late coactivators and transient coactivators.

The first part of our analysis highlights how to import ChIP-Seq data using the VULCAN adn DiffBind packages, provided alignment files (BAM format) and peak files (BED format) for each of the samples. In our dataset, we have 4 replicates for each time point.

Initial import

VULCAN requires an input sheet file in CSV format describing the samples and the location of the individual input files

```
sheetfile<-"chipseq/holding/sheet.csv"

fname<-"results/001_input.rda"
if(!file.exists(fname)){
  vobj<-vulcan.import(sheetfile)
  save(vobj,file=fname)
} else {
  load(fname)
}
```

Annotation

VULCAN identifies the location of each peak in each sample, and according to the selected method, assigns a score to each gene promoter. There are a few methods available.

- ***sum***: when multiple peaks are found, sum their contributions
- ***closest***: when multiple peaks are found, keep only the closest to the TSS as the representative one
- ***strongest***: when multiple peaks are found, keep the strongest as the representative one
- ***farthest***: when multiple peaks are found, keep only the closest to the TSS as the representative one
- ***topvar***: when multiple peaks are found, keep the most varying as the representative one
- ***lowvar***: when multiple peaks are found, keep the least varying as the representative one

```
vobj<-vulcan.annotate(vobj,lborder=-10000,rborder=10000,method="sum")
```

Normalization

At this step, genes are quantified according to the number of reads that could be associated to their promoters. The algorithms within VULCAN (*viper* and *DESeq2*) require however data to be normalized via Variance-Stabilizing Transformation (Anders and Huber, 2010).

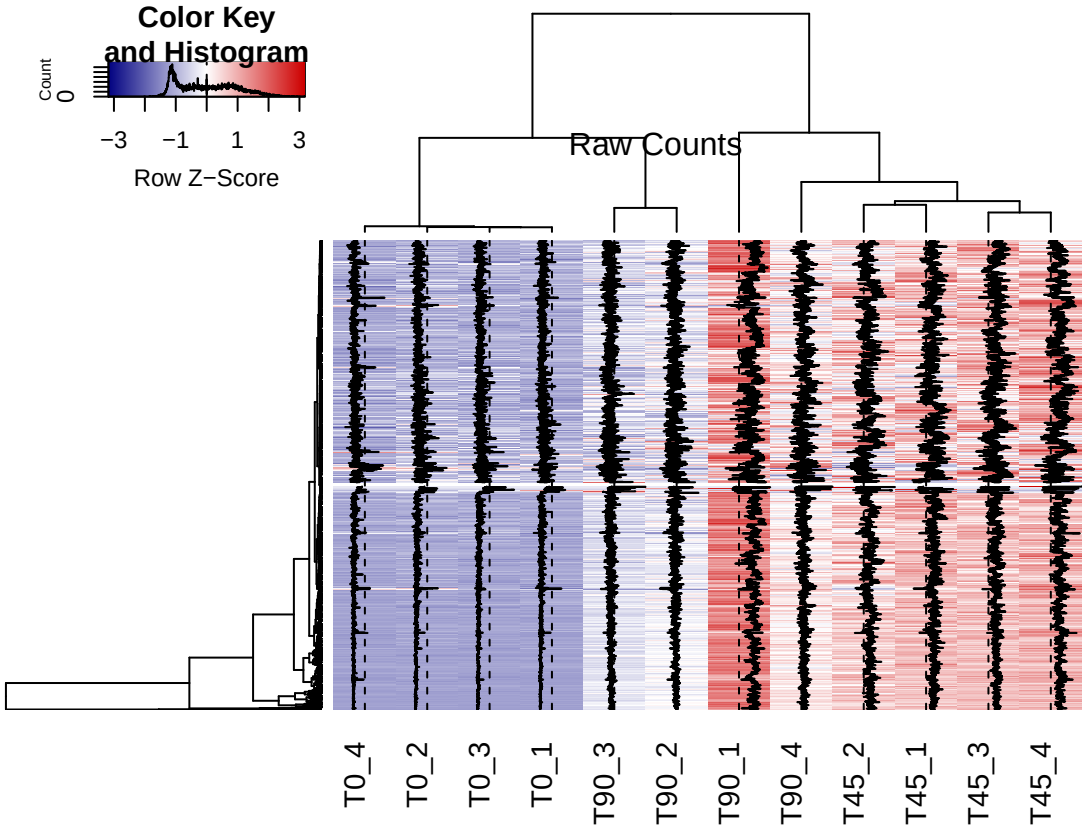


Figure 1: Figure S1. Dataset clustering with Raw Counts

```
vobj<-vulcan.normalize(vobj)
save(vobj,file="results/001_vobj.rda")
```

Dataset Sample Clustering

Here, we show how the samples cluster together using peak raw counts, VST-normalized peak raw counts and peak RPKMs. In the following heatmaps, there are 4288 rows, one per gene promoter, and 16 columns, one per sample. The sample name indicates the time point in minutes (T0, T45 or T90) and the number of replicate (_1, _2, _3, _4).

```
heatmap.2(vobj$rawcounts,scale="row",
          col=colorpanel(1000,"navy","white","red3"),tracecol="black",labRow="")
mtext("Raw Counts")
```

```
heatmap.2(vobj$normalized,
          col=colorpanel(1000,"navy","white","red3"),tracecol="black",labRow="")
mtext("VST-normalized")
```

```
heatmap.2(vobj$rpkms,scale="row",
          col=colorpanel(1000,"navy","white","red3"),tracecol="black",labRow="")
mtext("RPKMs")
```

Principal Component Analysis further confirms the clustering of replicates in two distinct groups: untreated (T0) and treated (T45/T90).

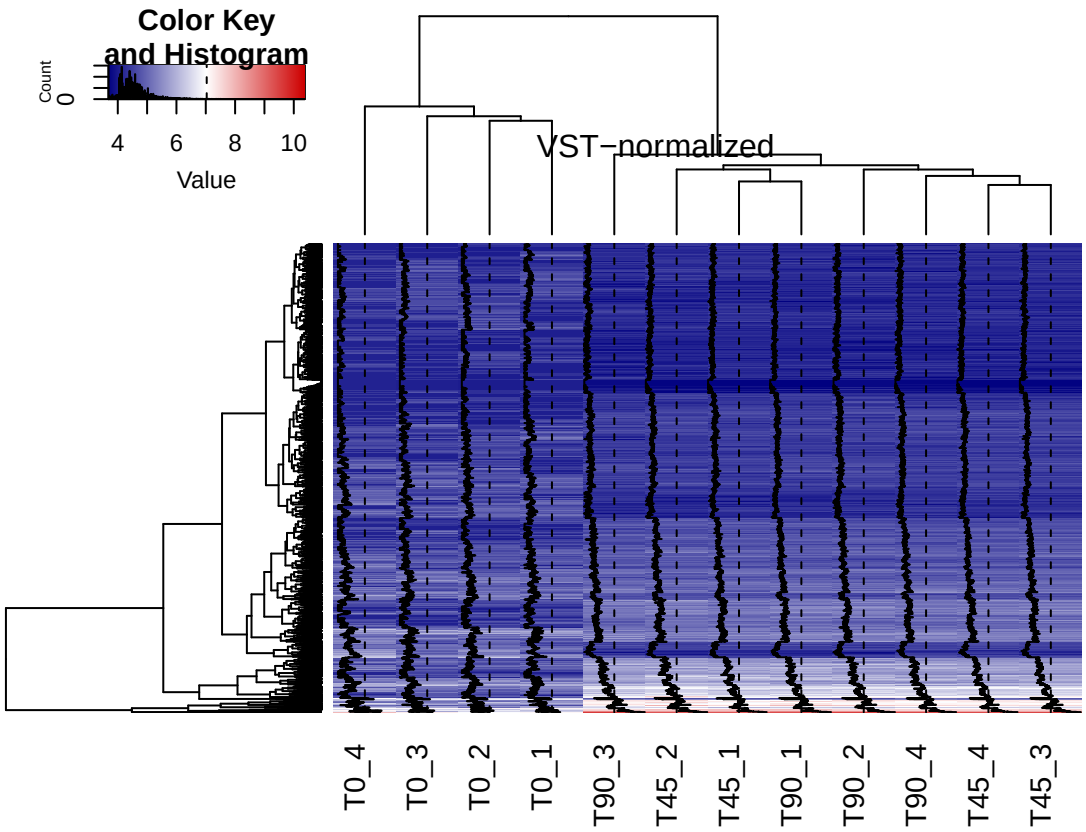


Figure 2: Figure S2. Dataset clustering with VST-normalized Counts

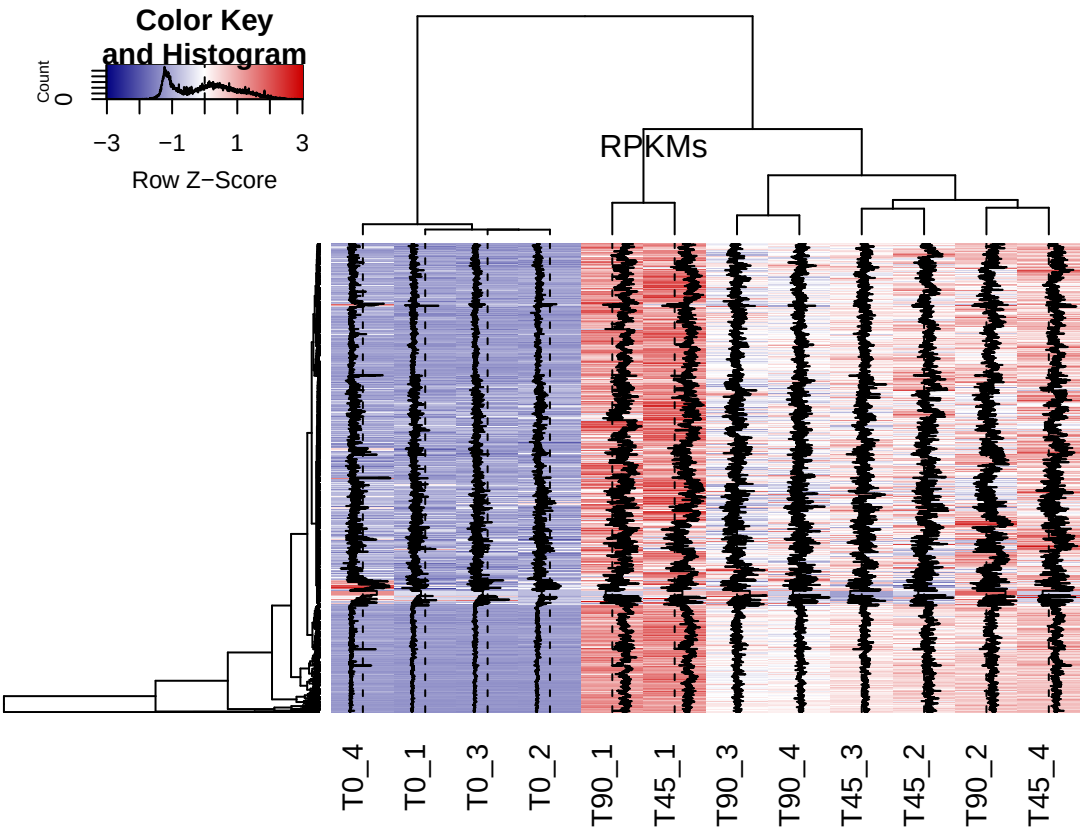


Figure 3: Figure S3. Dataset clustering with RPKMs

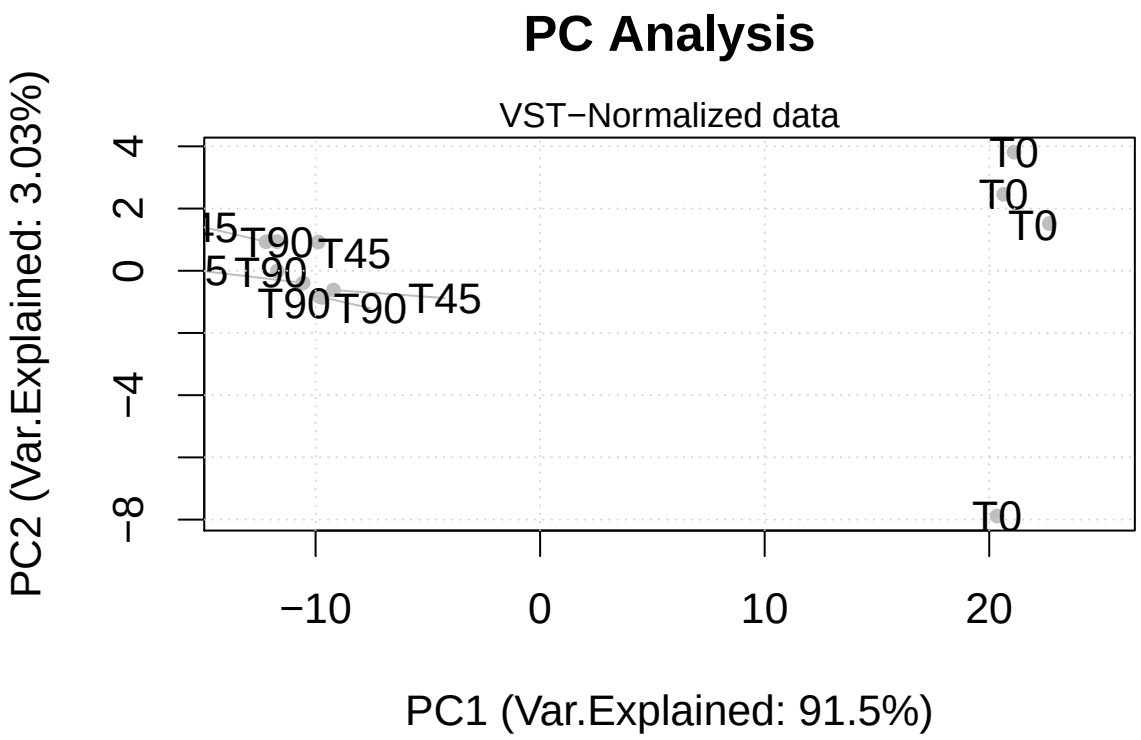


Figure 4: Figure S4. Principal Component Analysis of the dataset, highlighting components 1 and 2

```
topvar<-apply(vobj$normalized[,],1,var)
topvar<-sort(topvar,dec=TRUE)[1:500]
submat<-vobj$normalized[names(topvar),]
pca<-prcomp(t(submat))
vars<-100*(pca$sdev^2)/sum(pca$sdev^2)
vars<-signif(vars,3)

p1<-1
p2<-2
plot(pca$x[,p1],pca$x[,p2],
     xlab=paste0("PC",p1," (Var.Explained: ",vars[p1],"%)"),
     ylab=paste0("PC",p2," (Var.Explained: ",vars[p2],"%)"),
     type="p",main="PC Analysis",pch=20,col="grey",
     xlim=c(min(pca$x[,p1])*1.1,max(pca$x[,p1])*1.1)
)
mtext("VST-Normalized data",cex=0.8)
textplot2(pca$x[,p1],pca$x[,p2],new=FALSE,
          words=gsub("_[0-9]","",rownames(pca$x),perl=TRUE)
)
grid()
```

A clearer distinction of T45 and T90 is highlighted by PC5:

```
p1<-1
p2<-5
plot(pca$x[,p1],pca$x[,p2],
     xlab=paste0("PC",p1," (Var.Explained: ",vars[p1],"%)"),
     ylab=paste0("PC",p2," (Var.Explained: ",vars[p2],"%)"),
```

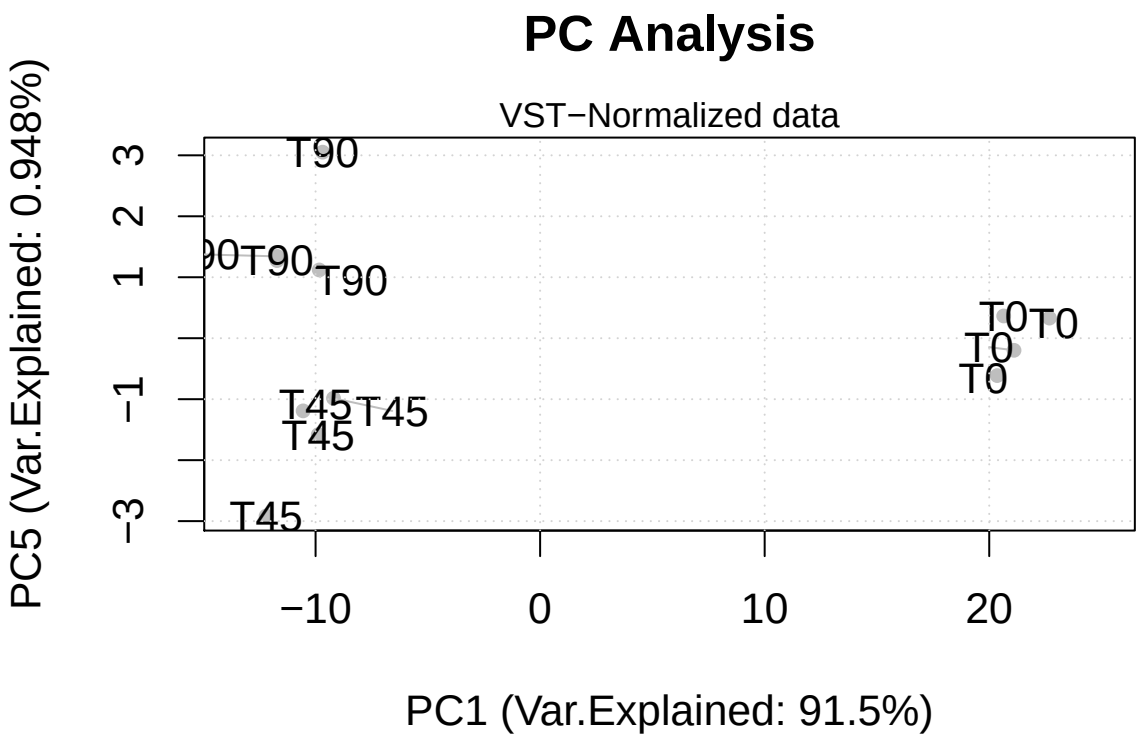


Figure 5: Figure S5. Principal Component Analysis of the dataset, highlighting components 1 and 5

```
     type="p",main="PC Analysis",pch=20,col="grey",
     xlim=c(min(pca$x[,p1])*1.1,max(pca$x[,p1])*1.1)
)
mtext("VST-Normalized data",cex=0.8)
textplot2(pca$x[,p1],pca$x[,p2],new=FALSE,
          words=gsub("_[0-9]","",rownames(pca$x),perl=TRUE)
)
grid()
```

MA plots

Without changing the format of the input data, we can use the Bioconductor DiffBind package to visualize the amplitude of changes in ER binding between time points. One way to do this is an MA plot, which shows the differences between measurements taken in two groups (e.g. 45' vs 00'), by transforming the promoter peak intensity data onto M (log ratio) and A (mean average) scales.

We will process the data using the DiffBind package and then use the *dba.plotMA* function to visualize the contrasts (which we can extract using the *dba.contrast* function).

```
sheetfile<-"chipseq/holding/sheet.csv"
fname<-"results/001_diffbind.rda"
if(!file.exists(fname)){
  # Load a sample sheet
  chipseqSamples<-read.csv(sheetfile)
  dbaobj<-dba(sampleSheet=chipseqSamples)

  # Count reads 500bp either side of summits, a peak is considered a peak
```

```
  # if it is found in 3 samples out of loaded samples.
  dbaobj<-dba.count(dbaobj,minOverlap=3,summits=500)

  # Define contrasts
  dbaobj<-dba.contrast(dbaobj, categories=DBA_CONDITION)

  # Calculate differential binding using the DESeq2 engine within DiffBind
  dbaobj<-dba.analyze(dbaobj,method=DBA_DESEQ2)

  # Save the DiffBind object
  save(dbaobj,file=fname)
} else {
  load(fname)
}

# Visualize each contrast
dba.contrast(dbaobj)
```

```
## 12 Samples, 19351 sites in matrix:
##       ID Tissue Factor Condition Replicate Caller Intervals FRiP
## 1  T90_1   MCF7     ER       t90         1 counts     19351 0.15
## 2  T90_2   MCF7     ER       t90         2 counts     19351 0.12
## 3  T90_3   MCF7     ER       t90         3 counts     19351 0.09
## 4  T90_4   MCF7     ER       t90         4 counts     19351 0.09
## 5  T45_1   MCF7     ER       t45         1 counts     19351 0.16
## 6  T45_2   MCF7     ER       t45         2 counts     19351 0.11
## 7  T45_3   MCF7     ER       t45         3 counts     19351 0.11
## 8  T45_4   MCF7     ER       t45         4 counts     19351 0.13
## 9   T0_1   MCF7     ER        t0         1 counts     19351 0.01
## 10  T0_2   MCF7     ER        t0         2 counts     19351 0.01
## 11  T0_3   MCF7     ER        t0         3 counts     19351 0.01
## 12  T0_4   MCF7     ER        t0         4 counts     19351 0.02
##
## 3 Contrasts:
##   Group1 Members1 Group2 Members2
## 1    t90        4    t45        4
## 2    t90        4     t0        4
## 3    t45        4     t0        4
```

```
dba.plotMA(dbaobj,contrast=1,method=DBA_DESEQ2)
```

```
dba.plotMA(dbaobj,contrast=2,method=DBA_DESEQ2)
```

```
dba.plotMA(dbaobj,contrast=3,method=DBA_DESEQ2)
```

Apply a coregulatory network over a ChIP-Seq profile

Once the data has been loaded, VULCAN applies a regulatory network over differential binding signatures to define Transcription Factors whose networks are most affected by the treatment. In our analysis, we will be using three different networks generated via ARACNe (Margolin et al., 2006): two are breast cancer-specific and are derived from the TCGA and METABRIC data collection, respectively. A third network is used as a negative control, highlighting regulatory mechanisms derived from the Amyloid Leukemia dataset (TCGA). ## Loading the input networks and processed data First, we will load the output from the previous

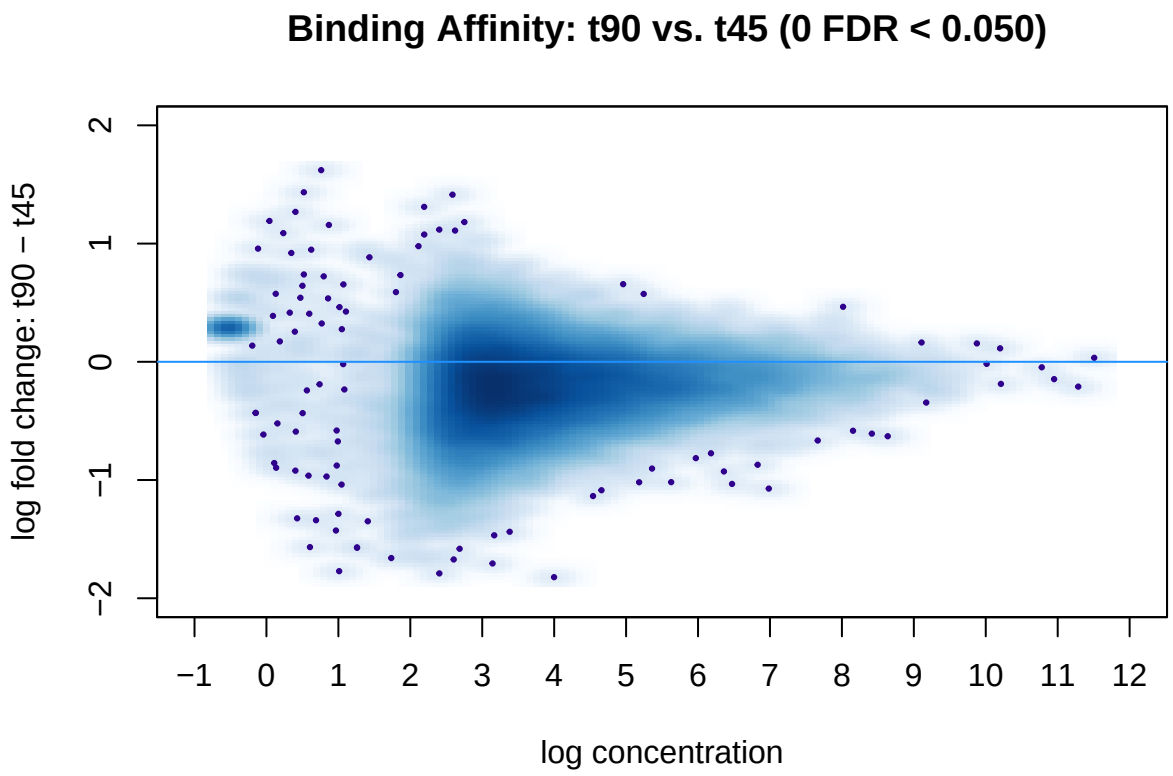


Figure 6: Figure S6. MA plot for Contrast 90mins vs 45mins, highlighting each individual peak. Significant peaks are highlighted in red

paragraphs (chipseq data annotated and normalized) and the three networks.

```
# Load imported vulcan object
load("results/001_vobj.rda")

# Vulcan Analysis (multiple networks)
## Networks
load("networks/laml-tf-regulon.rda")
laml_regulon<-regul
rm(regul)

load("networks/brca-tf-regulon.rda")
tcga_regulon<-regul
rm(regul)

load("networks/metabric-regulon-tfs.rda")
metabric_regulon<-regulon
rm(regulon)
```

Accessing the vulcan object /texit{vobj} can give us informations on the sample groups thereby contained.

```
names(vobj$samples)
```

```
## [1] "t90" "t45" "t0"
```

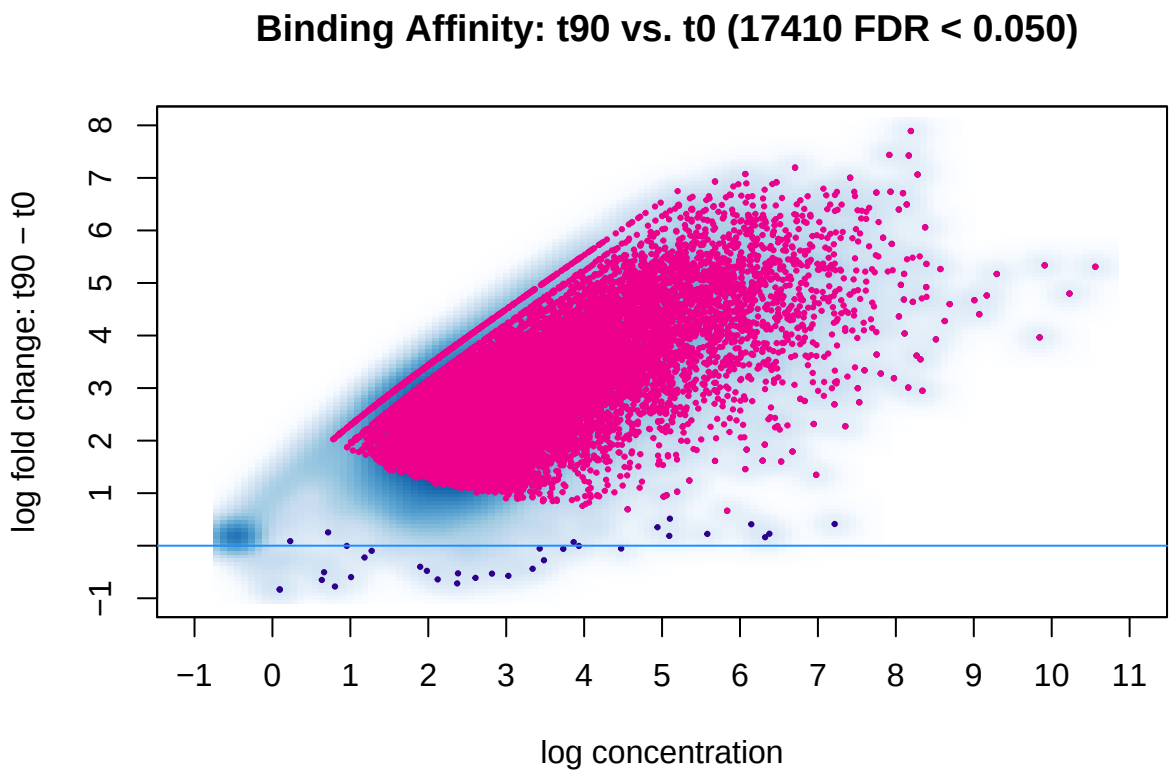


Figure 7: Figure S7. MA plot for Contrast 90mins vs 00mins, highlighting each individual peak. Significant peaks are highlighted in red

Running the core Vulcan function

The final Vulcan pipeline step requires three input objects:

- The annotated and normalized chipseq data (/texit{vobj})
- A specific binding signature defined by a contrast between two sample groups
- A regulatory network

```
fname<-"results/002_vobj_networks.rda"
list_eg2symbol<-as.list(org.Hs.egSYMBOL[mappedkeys(org.Hs.egSYMBOL)])
if(!file.exists(fname)){
  vobj_tcga_90<-vulcan(vobj,network=tcga_regulon,contrast=c("t90","t0"),annotation=list_eg2symbol)
  vobj_tcga_45<-vulcan(vobj,network=tcga_regulon,contrast=c("t45","t0"),annotation=list_eg2symbol)
  vobj_metabric_90<-vulcan(vobj,network=metabric_regulon,contrast=c("t90","t0"),annotation=list_eg2symbo
  vobj_metabric_45<-vulcan(vobj,network=metabric_regulon,contrast=c("t45","t0"),annotation=list_eg2symbo
  vobj_negative_90<-vulcan(vobj,network=laml_regulon,contrast=c("t90","t0"),annotation=list_eg2symbol)
  vobj_negative_45<-vulcan(vobj,network=laml_regulon,contrast=c("t45","t0"),annotation=list_eg2symbol)
  save(
    vobj_tcga_90,
    vobj_tcga_45,
    vobj_metabric_90,
    vobj_metabric_45,
    vobj_negative_90,
    vobj_negative_45,
    file=fname
  )
```

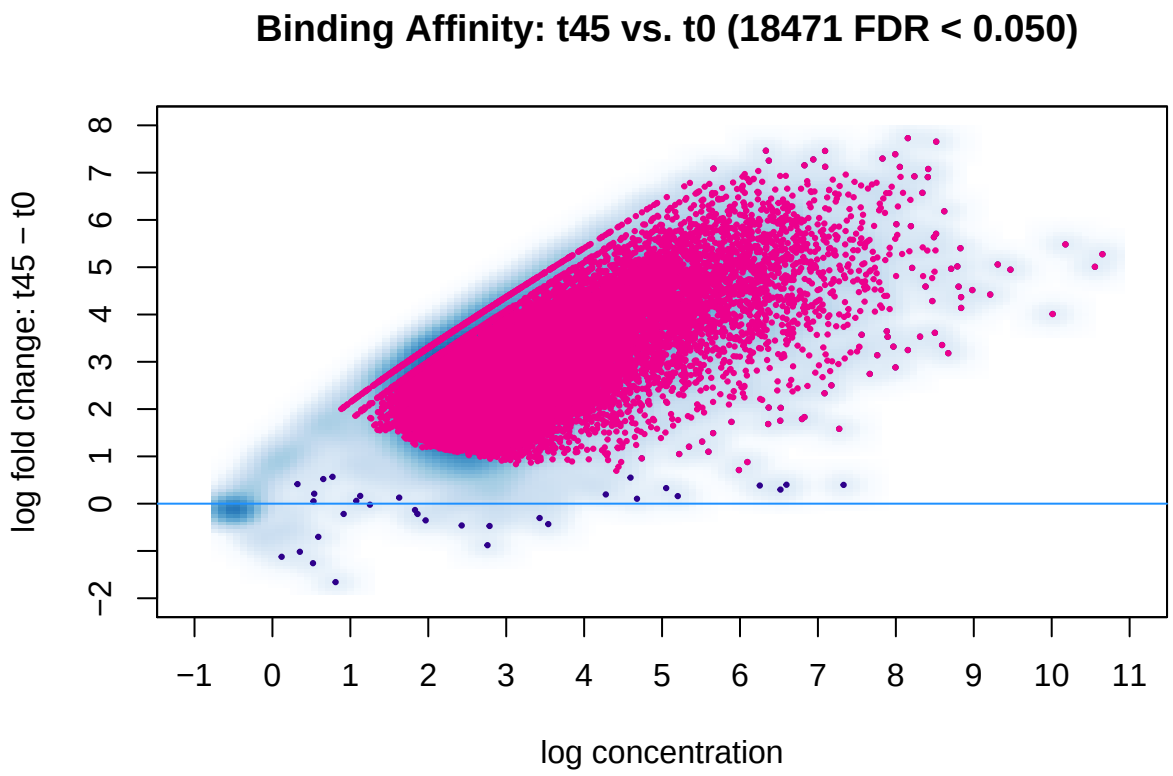


Figure 8: Figure S8. MA plot for Contrast 45mins vs 00mins, highlighting each individual peak. Significant peaks are highlighted in red

```
} else {
  load(fname)
}
```

Visualize the relative TF activity

Vulcan has now generated activity scores for each Transcription Factor, which specify the global ER binding strength on the promoters of their targets.

METABRIC results

The following line plot shows the relative network activity for every TF at two time points. On the y-axis, the Normalized Enrichment Score of Vulcan is provided. Dashed lines indicate a p=0.05 significance threshold for up- and down-regulation.

```
threshold<-p2z(0.05)
metabric_90=vobj_metabric_90$mrs[,"NES"]
metabric_45=vobj_metabric_45$mrs[,"NES"]
tfs<-names(metabric_45)
metabricmat<-cbind(rep(0,length(metabric_45)),metabric_45[tfs],metabric_90[tfs])
colnames(metabricmat)<-c("T0","T45","T90")
## All TFs
matplot(t(metabricmat),type="l",col="grey",ylab="VULCAN NES",xaxt="n",lty=1,main="All TFs",xlim=c(1,3.3
axis(1,at=c(1:3),labels=colnames(metabricmat))
```

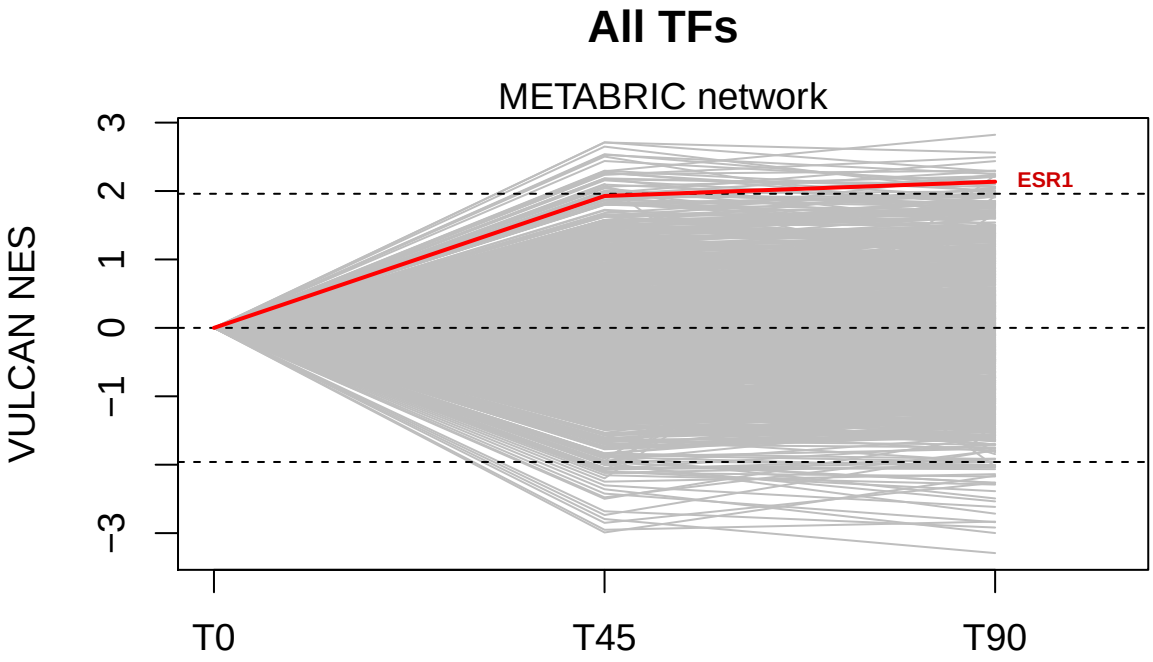


Figure 9: Figure S9. Global TF activity after Estradiol Treatment in MCF7 cells, inferred using the METABRIC network, highlighting the ESR1 TF as an example

```
abline(h=c(0,threshold,-threshold),lty=2)
matplot(t(t(metabricmat["ESR1",])),type="l",col="red",lty=1,lwd=2,add=TRUE)
text(3,metabricmat["ESR1",3],label="ESR1",pos=4,cex=0.6,font=2,col="red3")
mtext("METABRIC network")
```

These TFs can be grouped in classes. For example, TFs whose activity is already significant after 45mins and remains significant at 90 minutes after estradiol treatment can be dubbed **early coactivators**.

```
threshold<-p2z(0.05)
tfclass<-tfs[metabricmat[,"T45"]>=threshold&metabricmat[,"T90"]>=threshold]
matplot(t(metabricmat),type="l",col="grey",ylab="VULCAN NES",xaxt="n",lty=1,main="Early coactivators",x
axis(1,at=c(1:3),labels=colnames(metabricmat))
abline(h=c(0,threshold,-threshold),lty=2)
matplot(t(metabricmat[tfclass,]),type="l",col="red3",lty=1,lwd=2,add=TRUE)
# Repel a bit
plabels<-tfclass
oricoords<-sort(setNames(metabricmat[plabels,3],plabels))
newcoords<-setNames(seq(min(oricoords),max(oricoords),length.out=length(oricoords)),names(oricoords))
text(3,newcoords,label=names(oricoords),pos=4,cex=0.6,font=2)
text(3,newcoords["ESR1"],label="ESR1",pos=4,cex=0.6,font=2,col="red3")
mtext("METABRIC network")
```

TFs whose *repressed* targets are bound will yield a negative activity score. These **early corepressors** are symmetrically opposite to early responder TFs.

```
threshold<-p2z(0.05)
tfclass<-tfs[metabricmat[,"T45"]<=-threshold&metabricmat[,"T90"]<=-threshold]
matplot(t(metabricmat),type="l",col="grey",ylab="VULCAN NES",xaxt="n",lty=1,main="Early corepressors",x
axis(1,at=c(1:3),labels=colnames(metabricmat))
abline(h=c(0,threshold,-threshold),lty=2)
```

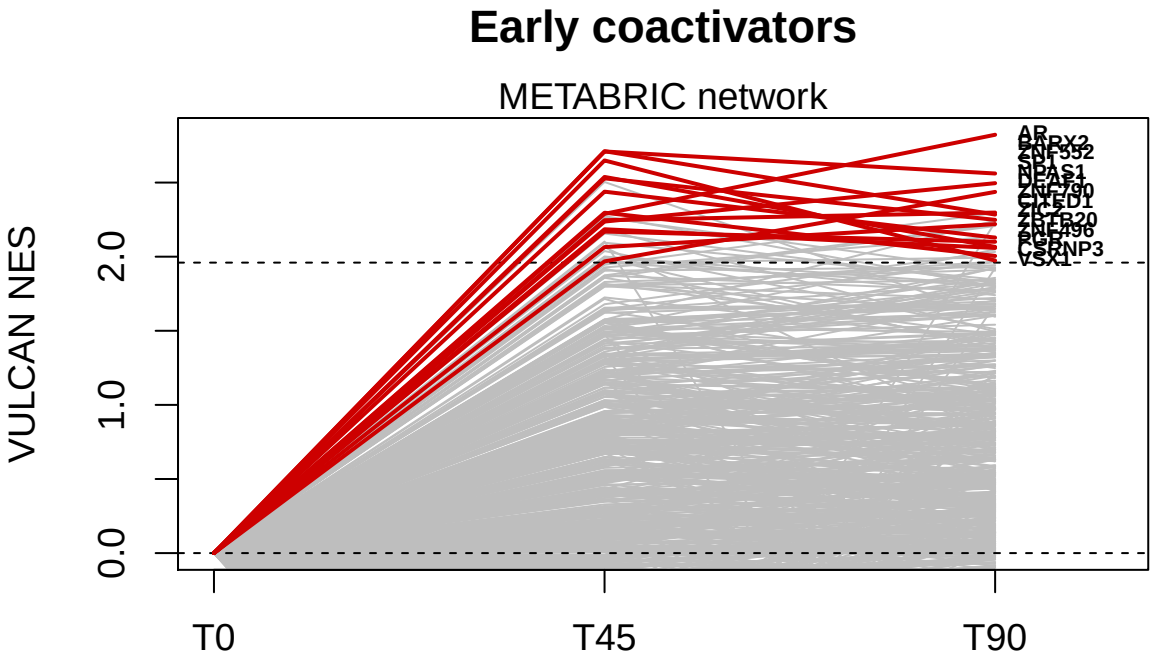


Figure 10: Figure S10. Global TF activity after Estradiol Treatment in MCF7 cells, inferred using the METABRIC network, highlighting TFs significantly upregulated at 45 minutes and 90 minutes

```
matplot(t(metabricmat[tfclass,]),type="l",col="cornflowerblue",lty=1,lwd=2,add=TRUE)
# Repel a bit
plabels<-c(tfclass,"GRHL2")
oricoords<-sort(setNames(metabricmat[plabels,3],plabels))
newcoords<-setNames(seq(min(oricoords),max(oricoords),length.out=length(oricoords)),names(oricoords))
text(3,newcoords,label=names(oricoords),pos=4,cex=0.6,font=2)
mtext("METABRIC network")
```

Some TFs appear to have their targets bound at 45 minutes, but then unoccupied at 90 minutes. This "updown" behavior is consistent to what observed in previous literature about the cyclic properties of certain components of the ER DNA-binding complex, and therefore we dubbed them **transient coactivators**.

```
threshold<-p2z(0.05)
tfclass<-tfs[metabricmat[,"T45"]>=threshold&metabricmat[,"T90"]<=threshold]
matplot(t(metabricmat),type="l",col="grey",ylab="VULCAN NES",xaxt="n",lty=1,main="Transient coactivators
axis(1,at=c(1:3),labels=colnames(metabricmat))
abline(h=c(0,threshold,-threshold),lty=2)
matplot(t(metabricmat[tfclass,]),type="l",col="seagreen",lty=1,lwd=2,add=TRUE)
# Repel a bit
plabels<-tfclass
oricoords<-sort(setNames(metabricmat[plabels,3],plabels))
newcoords<-setNames(seq(min(oricoords),max(oricoords),length.out=length(oricoords)),names(oricoords))
text(3,newcoords,label=names(oricoords),pos=4,cex=0.6,font=2)
mtext("METABRIC network")
```

Finally, a category of TFs appear to be activated but at 90 minutes only. We call this category of TFs **late coactivators**.

```
threshold<-p2z(0.05)
tfclass<-tfs[metabricmat[,"T45"]<=threshold&metabricmat[,"T45"]>0&metabricmat[,"T90"]>=threshold]
```

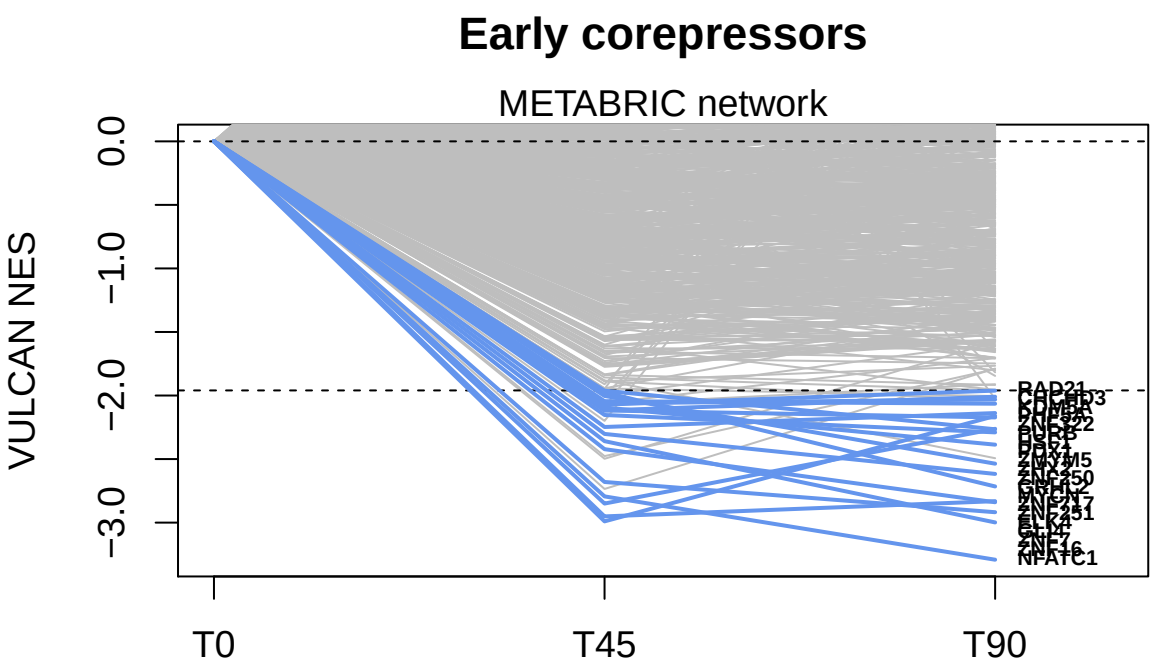


Figure 11: Figure S11. Global TF activity after Estradiol Treatment in MCF7 cells, inferred using the METABRIC network, highlighting TFs significantly downregulated at 45 minutes and 90 minutes

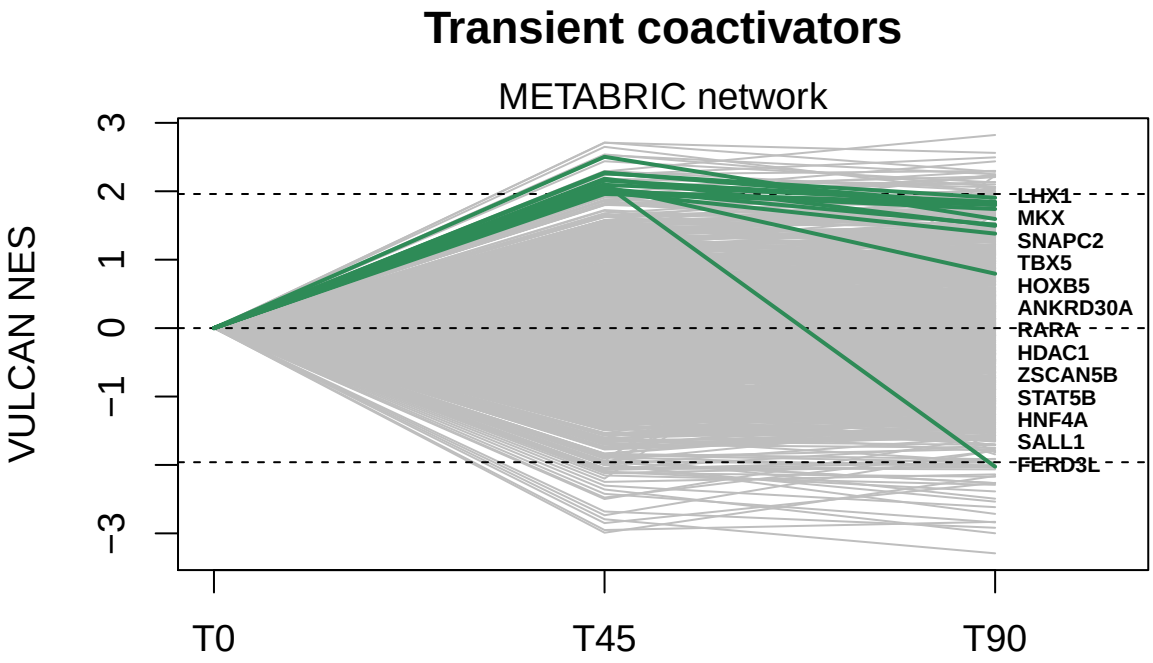


Figure 12: Figure S12. Global TF activity after Estradiol Treatment in MCF7 cells, inferred using the METABRIC network, highlighting TFs significantly upregulated at 45 minutes but not at 90 minutes

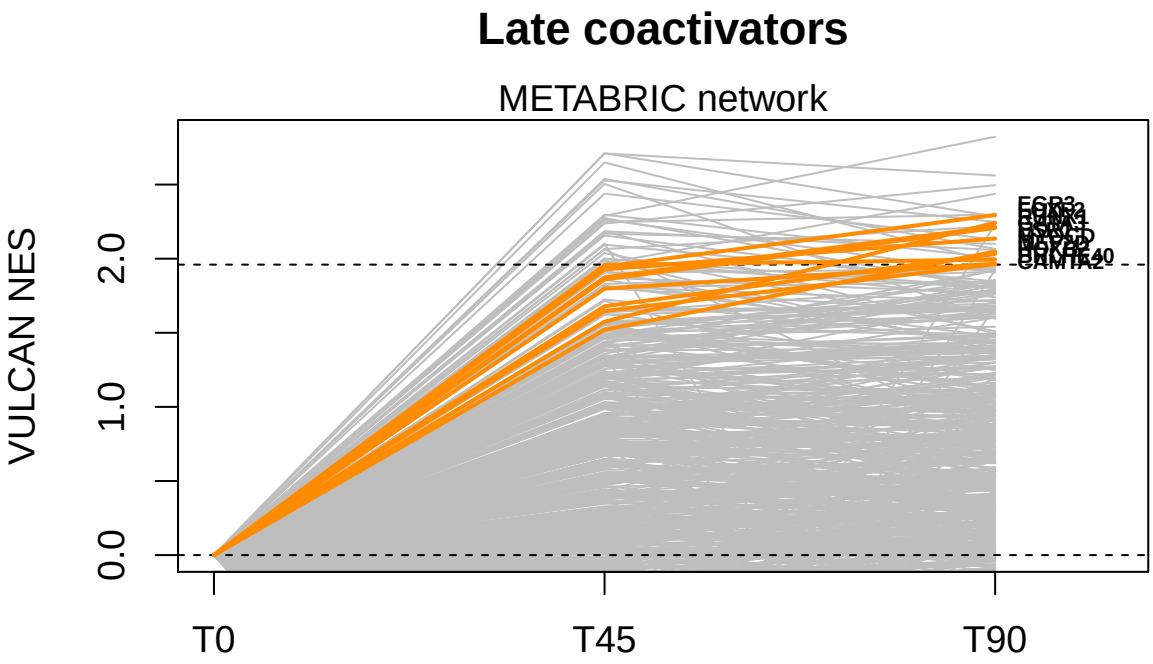


Figure 13: Figure S13. Global TF activity after Estradiol Treatment in MCF7 cells, inferred using the METABRIC network, highlighting TFs significantly upregulated at 90 minutes but not at 45 minutes

```
matplot(t(metabricmat),type="l",col="grey",ylab="VULCAN NES",xaxt="n",lty=1,main="Late coactivators",xli
axis(1,at=c(1:3),labels=colnames(metabricmat))
abline(h=c(0,threshold,-threshold),lty=2)
matplot(t(metabricmat[tfclass,]),type="l",col="darkorange",lty=1,lwd=2,add=TRUE)
# Repel a bit
plabels<-tfclass
oricoords<-sort(setNames(metabricmat[plabels,3],plabels))
newcoords<-setNames(seq(min(oricoords),min(oricoords)*1.2,length.out=length(oricoords)),names(oricoords)
text(3,newcoords,label=names(oricoords),pos=4,cex=0.6,font=2)
mtext("METABRIC network")
```

TCGA results

All the TFs and the four categories of TFs were inferred also using the TCGA network.

```
tcga_90=vobj_tcga_90$mrs[,"NES"]
tcga_45=vobj_tcga_45$mrs[,"NES"]
tfs<-names(tcga_45)
tcgamat<-cbind(rep(0,length(tcga_45)),tcga_45[tfs],tcga_90[tfs])
colnames(tcgamat)<-c("T0","T45","T90")
## All TFs
matplot(t(tcgamat),type="l",col="grey",ylab="VULCAN NES",xaxt="n",lty=1,main="All TFs",xlim=c(1,3.3),yli
axis(1,at=c(1:3),labels=colnames(tcgamat))
abline(h=c(0,threshold,-threshold),lty=2)
matplot(t(t(tcgamat["ESR1",])),type="l",col="red",lty=1,lwd=2,add=TRUE)
text(3,tcgamat["ESR1",3],label="ESR1",pos=4,cex=0.6,font=2,col="red3")
mtext("TCGA network")
```

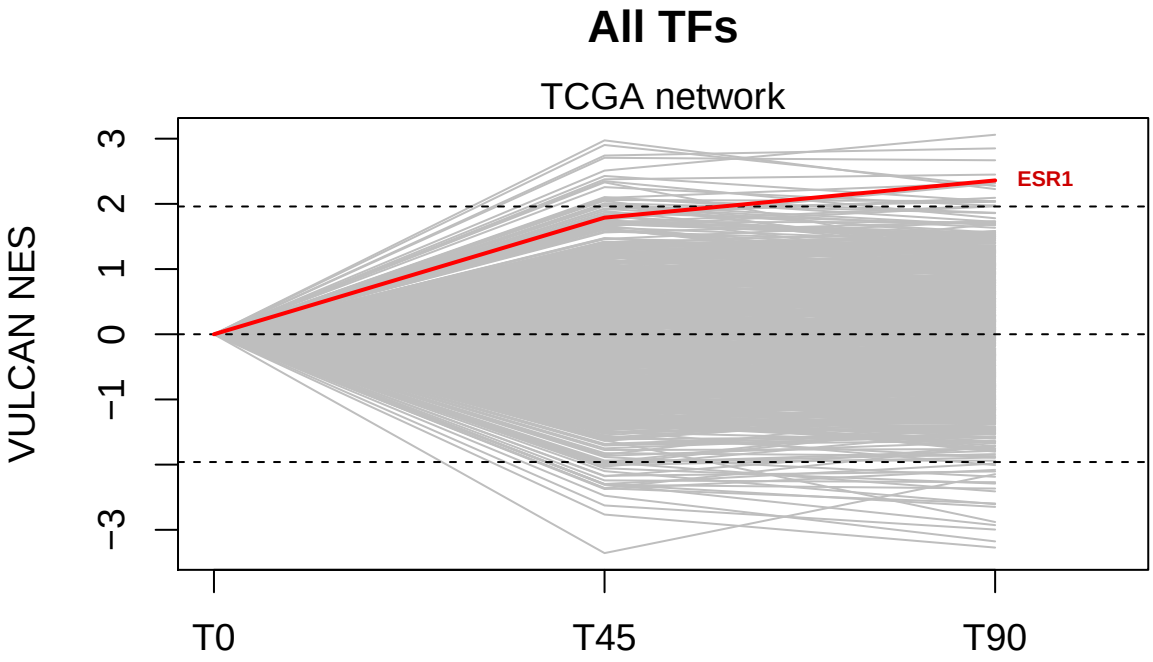


Figure 14: Figure S14. Global TF activity after Estradiol Treatment in MCF7 cells, inferred using the TCGA network, highlighting the ESR1 TF as an example

```
threshold<-p2z(0.05)
tfclass<-tfs[tcgamat[,"T45"]>=threshold&tcgamat[,"T90"]>=threshold]
matplot(t(tcgamat),type="l",col="grey",ylab="VULCAN NES",xaxt="n",lty=1,main="Early coactivators",xlim=
axis(1,at=c(1:3),labels=colnames(tcgamat))
abline(h=c(0,threshold,-threshold),lty=2)
matplot(t(tcgamat[tfclass,]),type="l",col="red3",lty=1,lwd=2,add=TRUE)
# Repel a bit
plabels<-tfclass
oricoords<-sort(setNames(tcgamat[plabels,3],plabels))
newcoords<-setNames(seq(min(oricoords),max(oricoords),length.out=length(oricoords)),names(oricoords))
text(3,newcoords,label=names(oricoords),pos=4,cex=0.6,font=2)
text(3,newcoords["ESR1"],label="ESR1",pos=4,cex=0.6,font=2,col="red3")
mtext("TCGA network")
```

```
threshold<-p2z(0.05)
tfclass<-tfs[tcgamat[,"T45"]<=-threshold&tcgamat[,"T90"]<=-threshold]
matplot(t(tcgamat),type="l",col="grey",ylab="VULCAN NES",xaxt="n",lty=1,main="Early corepressors",xlim=
axis(1,at=c(1:3),labels=colnames(tcgamat))
abline(h=c(0,threshold,-threshold),lty=2)
matplot(t(tcgamat[tfclass,]),type="l",col="cornflowerblue",lty=1,lwd=2,add=TRUE)
# Repel a bit
plabels<-tfclass
oricoords<-sort(setNames(tcgamat[plabels,3],plabels))
newcoords<-setNames(seq(min(oricoords),max(oricoords),length.out=length(oricoords)),names(oricoords))
text(3,newcoords,label=names(oricoords),pos=4,cex=0.6,font=2)
mtext("TCGA network")
```

```
threshold<-p2z(0.05)
tfclass<-tfs[tcgamat[,"T45"]>=threshold&tcgamat[,"T90"]<=threshold]
```

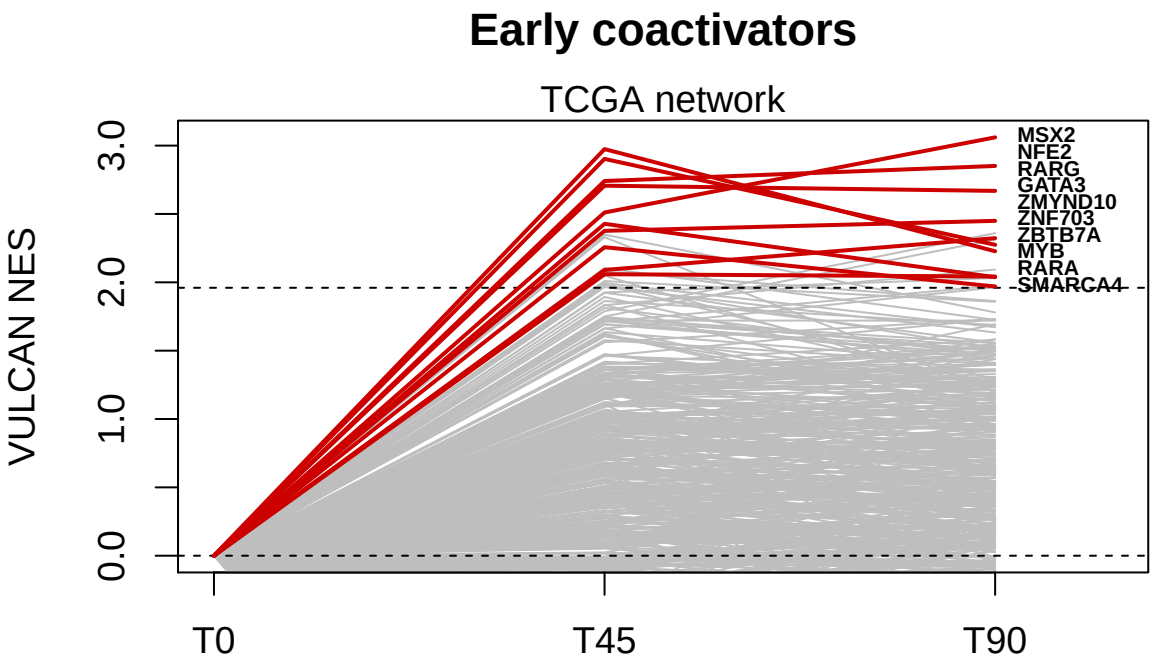


Figure 15: Figure S15. Global TF activity after Estradiol Treatment in MCF7 cells, inferred using the TCGA network, highlighting TFs significantly upregulated at 45 minutes and 90 minutes

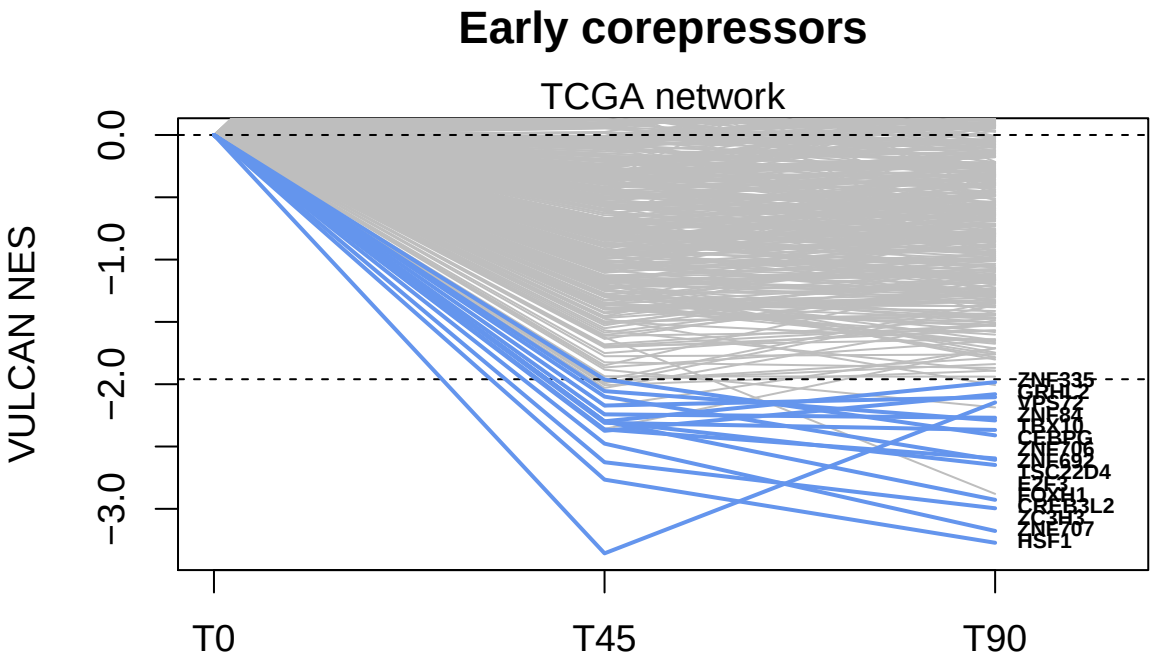


Figure 16: Figure S16. Global TF activity after Estradiol Treatment in MCF7 cells, inferred using the TCGA network, highlighting TFs significantly downregulated at 45 minutes and 90 minutes

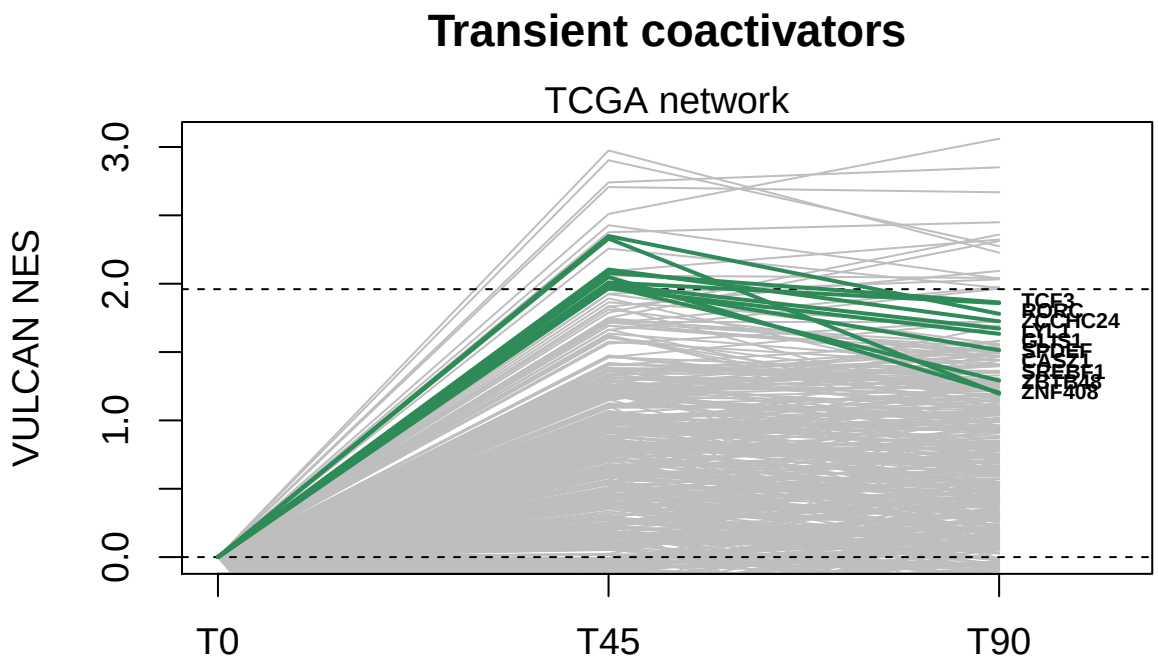


Figure 17: Figure S17. Global TF activity after Estradiol Treatment in MCF7 cells, inferred using the TCGA network, highlighting TFs significantly upregulated at 45 minutes but not at 90 minutes

```
matplot(t(tcgamat),type="l",col="grey",ylab="VULCAN NES",xaxt="n",lty=1,main="Transient coactivators",xl
axis(1,at=c(1:3),labels=colnames(tcgamat))
abline(h=c(0,threshold,-threshold),lty=2)
matplot(t(tcgamat[tfclass,]),type="l",col="seagreen",lty=1,lwd=2,add=TRUE)
# Repel a bit
plabels<-tfclass
oricoords<-sort(setNames(tcgamat[plabels,3],plabels))
newcoords<-setNames(seq(min(oricoords),max(oricoords),length.out=length(oricoords)),names(oricoords))
text(3,newcoords,label=names(oricoords),pos=4,cex=0.6,font=2)
mtext("TCGA network")
```

```
## Dn-Up TF class
threshold<-p2z(0.05)
tfclass<-tfs[tcgamat[,"T45"]<=threshold&tcgamat[,"T45"]>0&tcgamat[,"T90"]>=threshold]
matplot(t(tcgamat),type="l",col="grey",ylab="VULCAN NES",xaxt="n",lty=1,main="Late coactivators",xlim=c
axis(1,at=c(1:3),labels=colnames(tcgamat))
abline(h=c(0,threshold,-threshold),lty=2)
matplot(t(tcgamat[tfclass,]),type="l",col="darkorange",lty=1,lwd=2,add=TRUE)
# Repel a bit
plabels<-tfclass
oricoords<-sort(setNames(tcgamat[plabels,3],plabels))
newcoords<-setNames(seq(min(oricoords),min(oricoords)*1.2,length.out=length(oricoords)),names(oricoords)
text(3,newcoords,label=names(oricoords),pos=4,cex=0.6,font=2)
mtext("TCGA network")
```

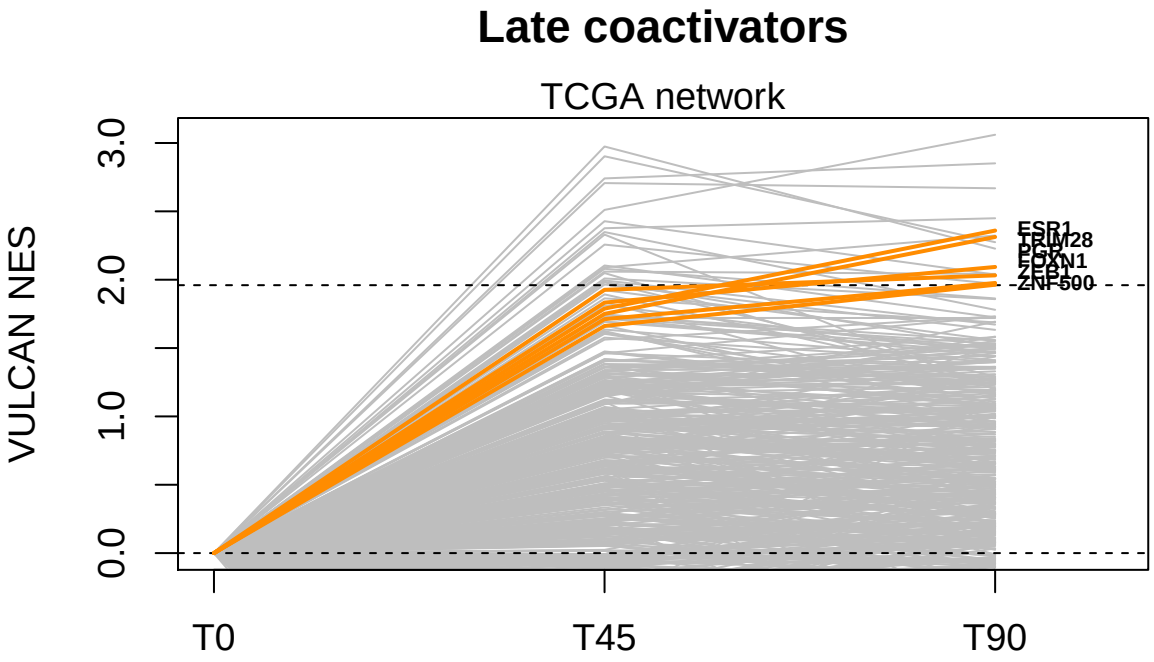


Figure 18: Figure S18. Global TF activity after Estradiol Treatment in MCF7 cells, inferred using the TCGA network, highlighting TFs significantly upregulated at 90 minutes but not at 45 minutes

Comparing TCGA and METABRIC results

The following scatterplots compare the activity inferred by VULCAN using two alternative networks, derived from the TCGA and METABRIC breast cancer datasets using ARACNe-AP (Giorgi et al., 2016) with default parameters.

```
common<-intersect(rownames(tcgamat),rownames(metabricmat))
set.seed(1)
x<-tcgamat[common,"T45"]
y<-metabricmat[common,"T45"]
plot(x,y,xlab="TCGA, 45mins",ylab="METABRIC, 45mins",pch=20,col="grey",xlim=c(min(x)*1.2,max(x)*1.2))
grid()
#toshow<-c("ESR1","GATA3","RARA","HSF1")
toshow<-names(which(rank(rank(-abs(x))+rank(-abs(y)))<=20))
#toshow<-unique(c(toshow,names(sort(rank(-abs(x))[1:10])),names(sort(rank(-abs(y)))[1:10])))
textplot2(x[toshow],y[toshow],words=toshow,new=FALSE,cex=0.8)
lm1<-lm(y~x)
abline(lm1$coef)
pcc<-cor.test(x,y)
mtext(paste0("PCC=",signif(pcc$estimate,2)," (p=",signif(pcc$p.value,2),")"))
```

```
x<-tcgamat[common,"T90"]
y<-metabricmat[common,"T90"]
plot(x,y,xlab="TCGA, 90mins",ylab="METABRIC, 90mins",pch=20,col="grey",xlim=c(min(x)*1.2,max(x)*1.2))
grid()
#toshow<-c("ESR1","GATA3","RARA","HSF1")
toshow<-names(which(rank(rank(-abs(x))+rank(-abs(y)))<=20))
#toshow<-unique(c(toshow,names(sort(rank(-abs(x))[1:10])),names(sort(rank(-abs(y)))[1:10])))
textplot2(x[toshow],y[toshow],words=toshow,new=FALSE,cex=0.8)
lm1<-lm(y~x)
```

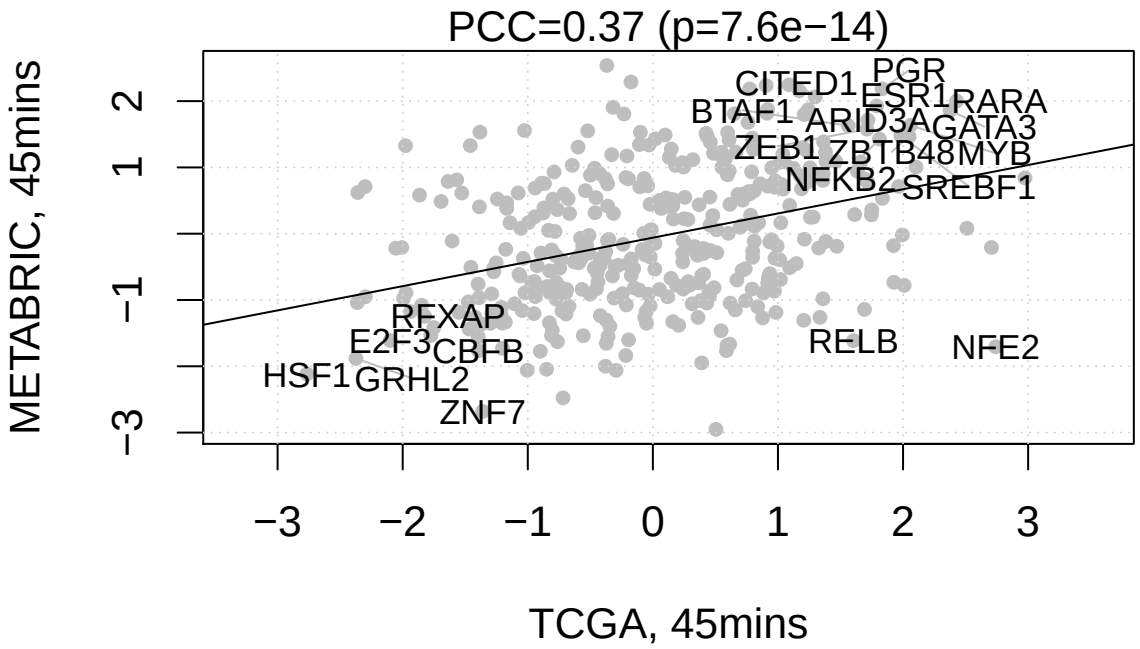


Figure 19: Figure S19. Comparing TF activities inferred by VULCAN using two different breast cancer dataset-derived networks on the ER 45 minutes signature. PCC indicates the Pearson Correlation Coefficient

```
abline(lm1$coef)
pcc<-cor.test(x,y)
mtext(paste0("PCC=",signif(pcc$estimate,2)," (p=",signif(pcc$p.value,2),")"))
abline(lm1$coef)
```

Testing with a different context network as a negative control

Regulatory networks can be tissue-specific, and so we built a third network using the same parameters as the two breast cancer data-based ones. The third network is built on leukemic samples from the TCGA AML dataset. Our results show a weaker activity for all TFs, which is only weakly correlated to that inferred via the TCGA breast cancer network.

```
negative_90=vobj_negative_90$mrs[,"NES"]
negative_45=vobj_negative_45$mrs[,"NES"]
tfs<-names(negative_45)
negativemat<-cbind(rep(0,length(negative_45)),negative_45[tfs],negative_90[tfs])
colnames(negativemat)<-c("T0","T45","T90")

common<-intersect(rownames(tcgamat),rownames(negativemat))

par(mfrow=c(1,2),oma=c(0,0,2,0))
# Scatterplot, negative vs tcga 45mins
x<-tcgamat[common,"T45"]
y<-negativemat[common,"T45"]
plot(x,y,xlab="TCGA, 45mins",ylab="negative, 45mins",pch=20,col="grey",xlim=c(min(x)*1.2,max(x)*1.2))
grid()
lm1<-lm(y~x)
abline(lm1$coef)
```

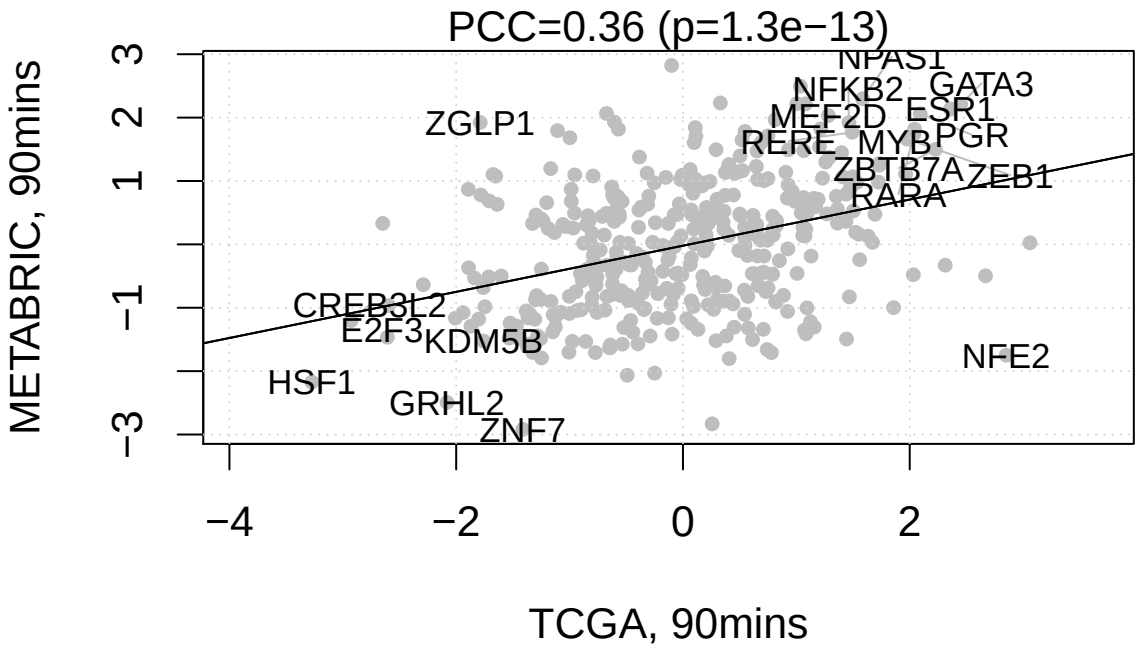


Figure 20: Figure S20. Comparing TF activities inferred by VULCAN using two different breast cancer dataset-derived networks on the ER 90 minutes signature. PCC indicates the Pearson Correlation Coefficient

```
pcc<-cor.test(x,y)
mtext(paste0("PCC=",signif(pcc$estimate,2)," (p=",signif(p.adjust(pcc$p.value,method="bonferroni",n=100

abline(lm1$coef)


# Scatterplot, negative vs tcga 90mins
x<-tcgamat[common,"T90"]
y<-negativemat[common,"T90"]
plot(x,y,xlab="TCGA, 90mins",ylab="negative, 90mins",pch=20,col="grey",xlim=c(min(x)*1.2,max(x)*1.2))
grid()
lm1<-lm(y~x)
abline(lm1$coef)
pcc<-cor.test(x,y)
mtext(paste0("PCC=",signif(pcc$estimate,2)," (p=",signif(p.adjust(pcc$p.value,method="bonferroni",n=100
abline(lm1$coef)

title("Comparison with Negative Network", outer=TRUE)
```

Pathway enrichment analysis

In the previous paragraphs we generated signatures of differential binding centered around the peaks extracted by VULCAN using the *vulcan.annotate* function. We apply two methods that assess the enrichment of gene pathways over continuous signatures: GSEA (Subramaniam et al., 2004) and aREA (Alvarez et al., 2016). First, we will load the VULCAN object containing our dataset:

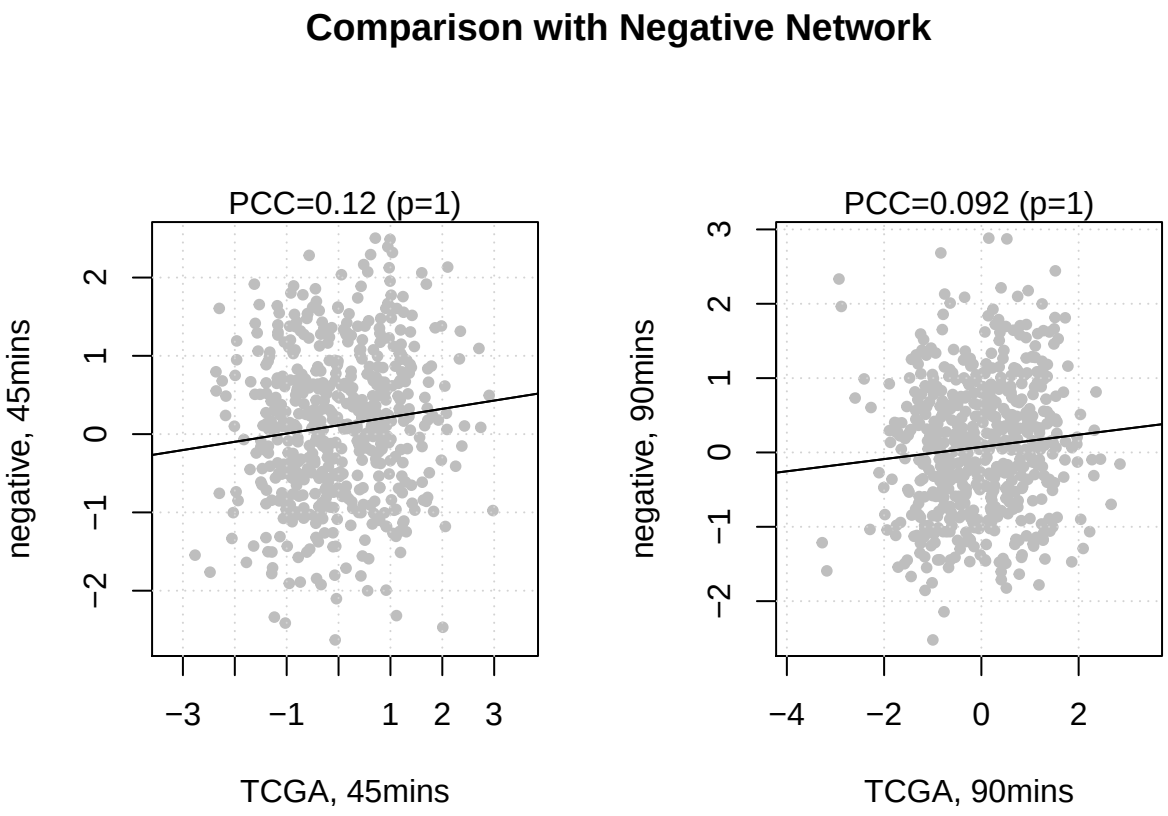


Figure 21: Figure S21. Comparison between activities inferred through a breast cancer TCGA dataset and the AML dataset. PCC indicates the Pearson Correlation Coefficient

```
# Load imported vulcan object
load("results/001_vobj.rda")
```

Then, we need gene pathways. A manually annotated and very exhaustive list is available from the MsiGDB database by the Broad Institute

```
# Load MsigDB
load("msigdb/MSigDB_v5.0_human.rda")
pathways<-msigDBentrez$c2.all.v5.0.entrez.gmt
```

First, we compare aREA and GSEA. Despite being similar in purpose, the two methods differ in their core algorithm and implementations. Notably, aREA processes the entire analysis in one go, by virtue of matrix multiplications. We expect aREA to be faster, while using more RAM than GSEA. Both GSEA and aREA are implemented in the VULCAN package. Here we calculate GSEA on the 90' vs 00' signature:

```
fname<-"results/003_pathwayComparison_GSEA_90.rda"
if(!file.exists(fname)){
  start<-Sys.time()
  results_gsea_90<-vulcan.pathways(vobj,pathways,contrast=c("t90","t0"),method="GSEA",np=1000)
  end<-Sys.time()
  time_gsea<-end-start
  save(results_gsea_90,time_gsea,file=fname)
} else {
  load(fname)
}
```

Then, we calculate aREA on the same 90' vs 00' signature:

```
fname<-"results/003_pathwayComparison_REA_90.rda"
if(!file.exists(fname)){
  start<-Sys.time()
  results_rea_90<-vulcan.pathways(vobj,pathways,contrast=c("t90","t0"),method="REA")[,1]
  end<-Sys.time()
  time_rea<-end-start
  save(results_rea_90,time_rea,file=fname)
} else {
  load(fname)
}
```

We then compare GSEA and aREA:

```
plot(results_rea_90,results_gsea_90,xlab="aREA enrichment score",ylab="GSEA enrichment score",pch=20,ma
     ylim=c(-max(abs(results_gsea_90)),max(abs(results_gsea_90))),col="grey")
th<-p2z(0.1)
abline(h=c(th,-th),v=c(th,-th),lty=2)
lm1<-lm(results_gsea_90~results_rea_90)
abline(lm1$coef,lwd=2)
pcc<-cor(results_rea_90,results_gsea_90,method="p")
mtext(paste0("PCC=",signif(pcc,3),", run on ",length(pathways)," pathways"))
# Estrogen
keywords<-c("ESTROGEN","ESTRADIOL","ESR1")
keypaths<-c()
for(keyword in keywords){
  keypaths<-c(keypaths,grep(keyword,names(pathways),value=TRUE))
}
keypaths<-unique(keypaths)
# Breast Cancer
```

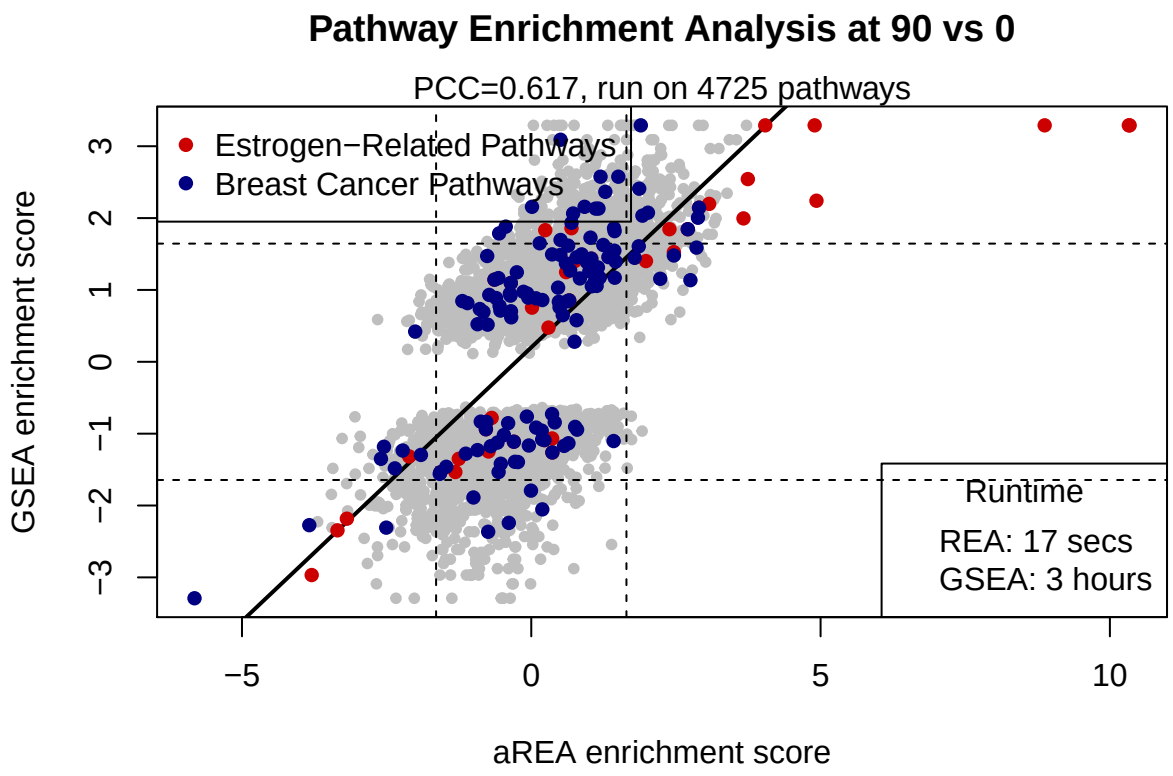


Figure 22: Figure S22. Comparison of GSEA and aREA on a differential binding signature

```
keywords<-c("BREAST_CANCER")
keypaths2<-c()
for(keyword in keywords){
  keypaths2<-c(keypaths2,grep(keyword,names(pathways),value=TRUE))
}
keypaths2<-unique(keypaths2)

points(results_rea_90[keypaths],results_gsea_90[keypaths],pch=16,col="red3")
points(results_rea_90[keypaths2],results_gsea_90[keypaths2],pch=16,col="navy")
legend("topleft",pch=16,col=c("red3","navy"),cex=1,legend=c("Estrogen-Related Pathways","Breast Cancer P
legend("bottomright",
       legend=c(
         paste0("REA: ",signif(time_rea,2)," ",units(time_rea)),
         paste0("GSEA: ",signif(time_gsea,2)," ",units(time_gsea))
       ),
       title="Runtime"
)
```

aREA appears to be faster. Therefore, we will use it to calculate the 45' signature as well:

```
fname<-"results/003_pathwayComparison_REA_45.rda"
if(!file.exists(fname)){
  start<-Sys.time()
  results_rea_45<-vulcan.pathways(vobj,pathways,contrast=c("t45","t0"),method="REA")[,1]
  end<-Sys.time()
  time_rea<-end-start
  save(results_rea_45,time_rea,file=fname)
```

```
} else {
  load(fname)
}
```

The results of aREA in both 90'vs00' and 45'vs00' signatures show the presence of known Estrogen-related pahtways:

```
topdn<-sort(results_rea_90)[1:20]
topup<-sort(results_rea_90,dec=TRUE)[1:20]

topdn<-cbind(topdn,z2p(topdn))
topup<-cbind(topup,z2p(topup))
colnames(topdn)<-colnames(topup)<-c("NES, 90' vs 0'","pvalue")

rownames(topdn)<-tolower(rownames(topdn))
rownames(topup)<-tolower(rownames(topup))
rownames(topdn)<-gsub("_"," ",rownames(topdn))
rownames(topup)<-gsub("_"," ",rownames(topup))
topdn[,1]<-signif(topdn[,1],3)
topup[,1]<-signif(topup[,1],3)
topdn[,2]<-signif(topdn[,2],2)
topup[,2]<-signif(topup[,2],2)

firstup <- function(x) {
  substr(x, 1, 1) <- toupper(substr(x, 1, 1))
  x
}
rownames(topdn)<-firstup(rownames(topdn))
rownames(topup)<-firstup(rownames(topup))
```

```
# Plot your table with table Grob in the library(gridExtra)
grid.newpage()
grid.table(topup)
```

```
grid.newpage()
grid.table(topdn)
```

```
topdn<-sort(results_rea_45)[1:20]
topup<-sort(results_rea_45,dec=TRUE)[1:20]

topdn<-cbind(topdn,z2p(topdn))
topup<-cbind(topup,z2p(topup))
colnames(topdn)<-colnames(topup)<-c("NES, 45' vs 0'","pvalue")

rownames(topdn)<-tolower(rownames(topdn))
rownames(topup)<-tolower(rownames(topup))
rownames(topdn)<-gsub("_"," ",rownames(topdn))
rownames(topup)<-gsub("_"," ",rownames(topup))
topdn[,1]<-signif(topdn[,1],3)
topup[,1]<-signif(topup[,1],3)
topdn[,2]<-signif(topdn[,2],2)
topup[,2]<-signif(topup[,2],2)

firstup <- function(x) {
  substr(x, 1, 1) <- toupper(substr(x, 1, 1))
```

Bhat esr1 targets via akt1 up	10.3	5.3e−25
Dutertre estradiol response 6hr up	8.87	7.1e−19
Frasor response to estradiol up	4.93	8.3e−07
Dutertre estradiol response 24hr up	4.9	9.7e−07
Stein esr1 targets	4.04	5.3e−05
Stein esrra targets responsive to estrogen dn	3.74	0.00018
Creighton endocrine therapy resistance 1	3.72	2e−04
Stossi response to estradiol	3.67	0.00024
Zwang egf interval dn	3.55	0.00039
Creighton endocrine therapy resistance 4	3.42	0.00062
Pedrioli mir31 targets up	3.38	0.00071
Massarweh tamoxifen resistance dn	3.24	0.0012
Lein pons markers	3.19	0.0014
Reactome hs gag degradation	3.19	0.0014
Jiang tip30 targets dn	3.17	0.0015
Kegg glycerophospholipid metabolism	3.16	0.0016
Geserick tert targets dn	3.13	0.0018

Figure 23: Table S2. aREA results: upregulated MsigDBpathways at 90mins

Nikolsky breast cancer 1q21 amplicon	−3.84	0.000
Dutertre estradiol response 24hr dn	−3.79	0.000
Streicher lsm1 targets up	−3.69	0.000
Kenny ctnnb1 targets dn	−3.45	0.000
Reactome activation of chaperone genes by xbp1s	−3.44	0.000
Verrecchia response to tgfb1 c4	−3.43	0.000
Frasor response to estradiol dn	−3.35	0.000
Pid myc repress pathway	−3.27	0.001
Sesto response to uv c4	−3.19	0.001
Dutertre estradiol response 6hr dn	−3.19	0.001
Reactome degradation of the extracellular matrix	−3.17	0.001
Lamb ccnd1 targets	−3.05	0.002
Guo targets of irs1 and irs2	−3.04	0.002
ıbayashi adipogenesis pparg rxra bound with h4k20me1 mark	−2.98	0.002
Woo liver cancer recurrence up	−2.96	0.003
Pid myc pathway	−2.95	0.003
Hu angiogenesis dn	−2.89	0.003

Figure 24: Table S3. aREA results: downregulated MsigDBpathways at 90mins

Bhat esr1 targets not via akt1 up	10.4	2.8e−25
Dutertre estradiol response 6hr up	8.1	5.5e−16
Frasor response to estradiol up	4.54	5.6e−06
Stein esrra targets responsive to estrogen dn	4.21	2.6e−05
Dutertre estradiol response 24hr up	4.15	3.4e−05
Stein esr1 targets	4.03	5.6e−05
Vantveer breast cancer esr1 up	3.84	0.00012
Geserick tert targets dn	3.69	0.00022
Pedrioli mir31 targets up	3.61	3e−04
Reactome glycerophospholipid biosynthesis	3.39	0.00069
Zwang egf interval dn	3.38	0.00072
Naba ecm affiliated	3.31	0.00093
Stossi response to estradiol	3.29	0.00099
Lien breast carcinoma metaplastic vs ductal dn	3.25	0.0011
Kegg glycerophospholipid metabolism	3.25	0.0012
Creighton endocrine therapy resistance 1	3.24	0.0012
Reactome hs gag degradation	3.21	0.0013

Figure 25: Table S5. aREA results: upregulated MsigDBpathways at 45mins

```
  x
}
rownames(topdn)<-firstup(rownames(topdn))
rownames(topup)<-firstup(rownames(topup))
```

```
# Plot your table with table Grob in the library(gridExtra)
grid.newpage()
grid.table(topup)
```

```
grid.newpage()
grid.table(topdn)
```

The GRHL2 Transcription Factor

Our analysis shows that the genes repressed by the Transcription Factor GRHL2 are occupied by the ER complex, using both TCGA-derived and METABRIC-derived regulatory models.

Network similarity between ESR1 and GRHL2

The following analysis highlights, with a Venn Diagram for both the METABRIC and TCGA ARACNe-derived regulatory models, the overlap between the ESR1 (Estrogen Receptor) and GRHL2 networks

Dutertre estradiol response 24hr dn	−3.97	7.2e−05
Kenny ctnnb1 targets dn	−3.89	1e−04
Nikolsky breast cancer 1q21 amplicon	−3.8	0.00014
Streicher lsm1 targets up	−3.7	0.00021
Wakabayashi adipogenesis pparg rxra bound 8d	−3.45	0.00055
Boyault liver cancer subclass g23 up	−3.16	0.0016
Reactome activation of chaperone genes by xbp1s	−3.13	0.0017
Dutertre estradiol response 6hr dn	−3.04	0.0023
Sesto response to uv c4	−3.01	0.0026
Shin b cell lymphoma cluster 2	−2.96	0.003
Yamashita liver cancer with epcam dn	−2.91	0.0037
Ichiba graft versus host disease 35d up	−2.82	0.0048
Chen hoxa5 targets 9hr up	−2.77	0.0057
Kenny ctnnb1 targets up	−2.74	0.0061
Kegg sulfur metabolism	−2.72	0.0065
Laiho colorectal cancer serrated up	−2.72	0.0066
Reactome unfolded protein response	−2.68	0.0074

Figure 26: Table S6. aREA results: downregulated MsigDBpathways at 45mins

```
# Load networks
load("networks/brca-tf-regulon.rda")
tcga_regulon<-regul
rm(regul)

load("networks/metabric-regulon-tfs.rda")
metabric_regulon<-regulon
rm(regulon)


# Select TFs
tf1<-"ESR1"
tf2<-"GRHL2"


par(mfrow=c(1,2))

# Venn diagram in TCGA
reg1<-tcga_regulon[[list_symbol2eg[[tf1]]]]
reg2<-tcga_regulon[[list_symbol2eg[[tf2]]]]
targets1<-names(reg1$tfmode)
targets2<-names(reg2$tfmode)
vennlist<-list()
vennlist[[tf1]]<-targets1
vennlist[[tf2]]<-targets2
venn(vennlist)
title("Network overlap in TCGA")


# Venn diagram in METABRIC
reg1<-metabric_regulon[[list_symbol2eg[[tf1]]]]
reg2<-metabric_regulon[[list_symbol2eg[[tf2]]]]
targets1<-names(reg1$tfmode)
targets2<-names(reg2$tfmode)
vennlist<-list()
vennlist[[tf1]]<-targets1
vennlist[[tf2]]<-targets2
venn(vennlist)
title("Network overlap in METABRIC")
```

```
par<-par.backup
```

Correlation between ESR1 and GRHL2 expression in breast cancer datasets

The following analysis assesses the inverse correlation between GRHL2 and ESR1 expressions in TCGA and METABRIC, and highlights the phenomenon for specific tumor subtypes, according to the PAM50 nomenclature (Perou et al., 2000)

```
# TCGA Color annotation
load("data/brca-expmat.rda")
load("data/brca-subtypes.rda")
expmat<-expmat[,names(subtypes)]
colors<-setNames(rep("black",length(subtypes)),subtypes)
```

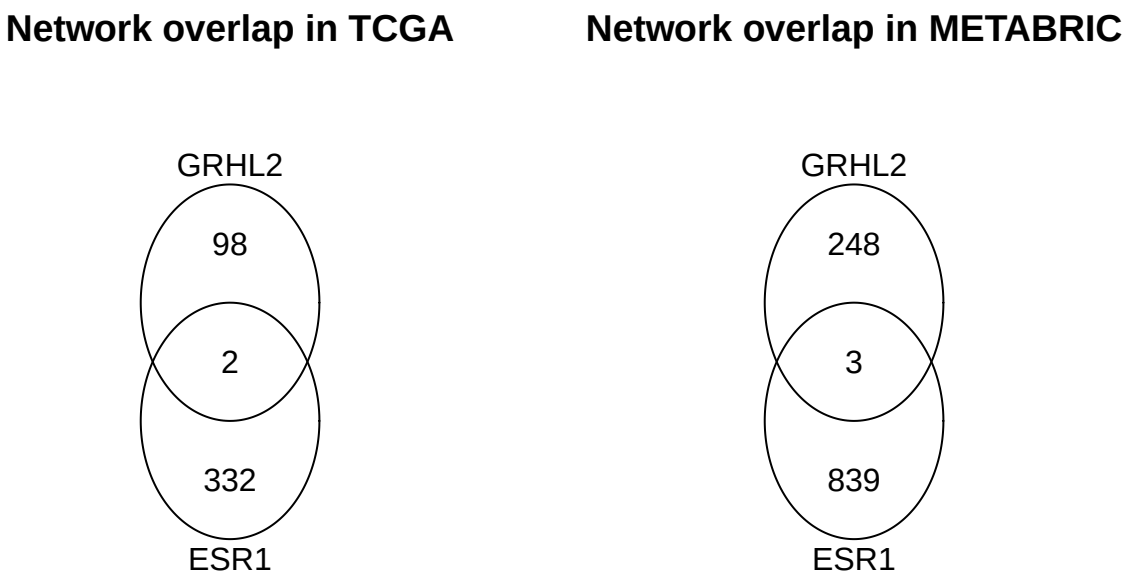


Figure 27: Figure S23. Overlap between ESR1 and GRH2 networks in TCGA (left) and METABRIC (right) networks

```
colors[subtypes=="Basal"]<-"#FF0000AA"
colors[subtypes=="LumA"]<-"#00FFFFAA"
colors[subtypes=="LumB"]<-"#0000FFAA"
colors[subtypes=="Her2"]<-"#FFFF00AA"
colors[subtypes=="Tumor, Normal-Like"]<-"#00FF00AA"
colors[subtypes=="Normal, Tumor-Like"]<-"#00FF00AA"
colors[subtypes=="Normal"]<-"#00FF00AA"
colors_tcga<-colors
tcga_expmat<-expmat

# METABRIC Color annotation
load("data/metabric-expmat.rda")
load("data/metabric-subtypes.rda")
expmat<-expmat[,names(subtypes)]
colors<-setNames(rep("black",length(subtypes)),subtypes)
colors[subtypes=="Basal"]<-"#FF0000AA"
colors[subtypes=="LumA"]<-"#00FFFFAA"
colors[subtypes=="LumB"]<-"#0000FFAA"
colors[subtypes=="Her2"]<-"#FFFF00AA"
colors[subtypes=="Tumor, Normal-Like"]<-"#00FF00AA"
colors[subtypes=="Normal, Tumor-Like"]<-"#00FF00AA"
colors[subtypes=="Normal"]<-"#00FF00AA"
colors_metabric<-colors
metabric_expmat<-expmat

x<-tcga_expmat[list_symbol2eg[[tf1]],]
y<-tcga_expmat[list_symbol2eg[[tf2]],]
plot(x,y,xlab=tf1,ylab=tf2,pch=20,main="Correlation between GRHL2 and ESR1 expression in TCGA",col=colo
pcc<-cor(x,y)
mtext(paste0("PCC=",signif(pcc,3)))
grid()
```

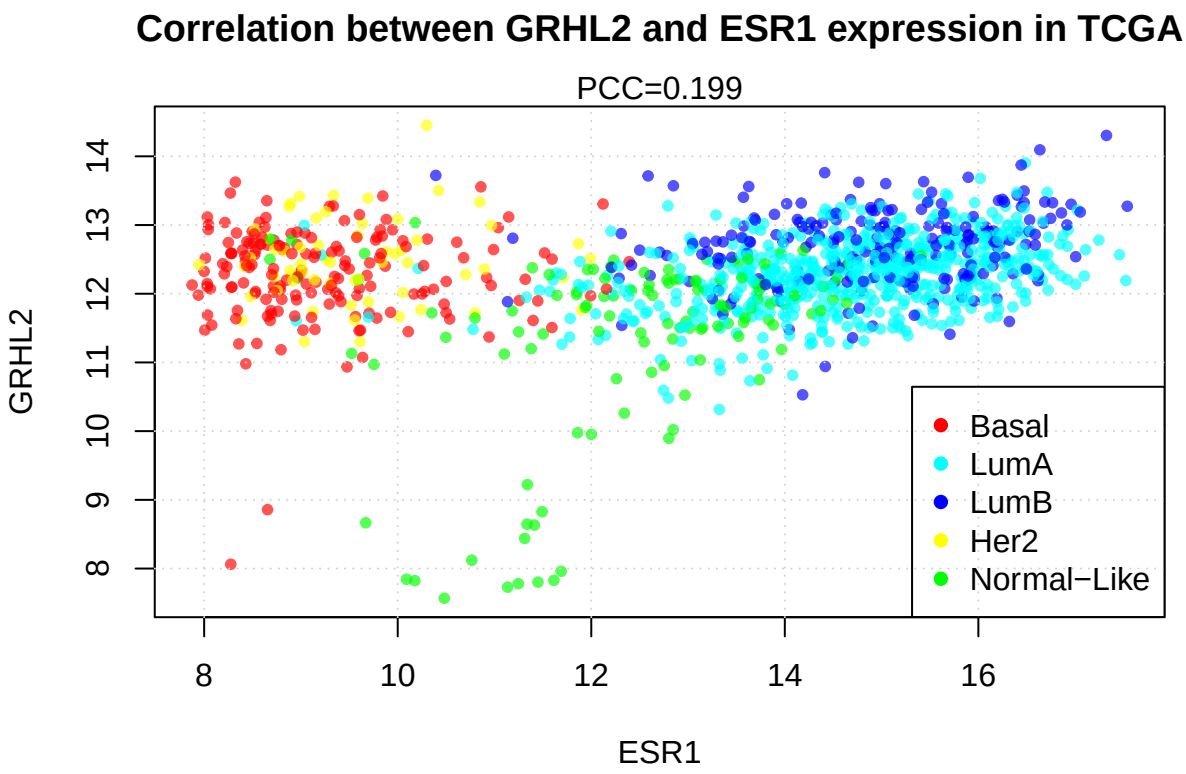
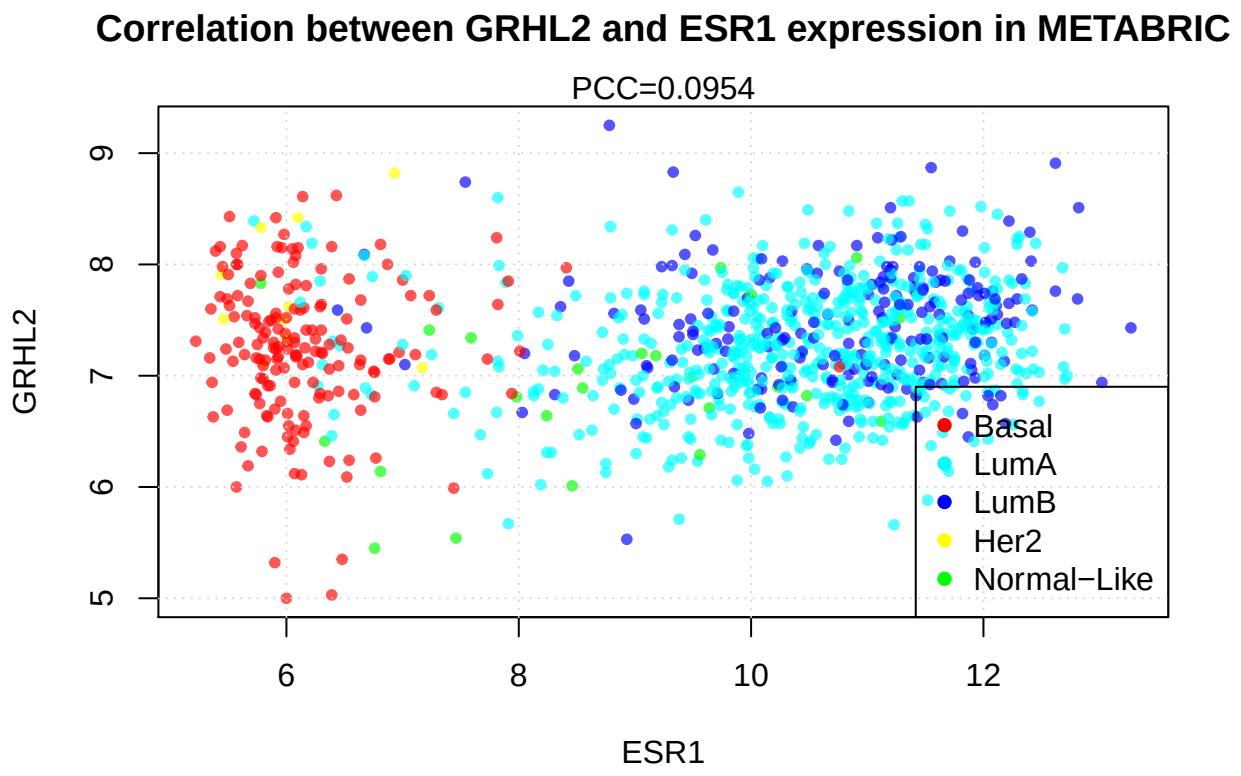


Figure 28: Figure S24. Correlation between GRHL2 and ESR1 expression in the TCGA breast cancer dataset

```
legend("bottomright",col=c("red","cyan","blue","yellow","green"),legend=c("Basal","LumA","LumB","Her2",
```

```
x<-metabric_expmat[list_symbol2eg[[tf1]],]
y<-metabric_expmat[list_symbol2eg[[tf2]],]
plot(x,y,xlab=tf1,ylab=tf2,pch=20,main="Correlation between GRHL2 and ESR1 expression in METABRIC",col=
pcc<-cor(x,y)
mtext(paste0("PCC=",signif(pcc,3)))
grid()
legend("bottomright",col=c("red","cyan","blue","yellow","green"),legend=c("Basal","LumA","LumB","Her2",
```



Enrichment of GRHL2 network for gene pathways

In order to analyze the GRHL2 network, we perform a simple hypergeometric test to overlap its targets with all pathways contained in hte manually curated Reactome collection of MsigDB pathways (C2 v5.0).

```
tf<-"GRHL2"
regulon<-tcga_regulon

# Pathways
wass<-load("msigdb/MSigDB_v5.0_human.rda") # "msigDBentrez" "msigDBsymbol"
ngenes<-length(unique(unlist(msigDBentrez$c2.all.v5.0.entrez.gmt)))

### Hypergeometric test implementation
# Hypergeometric enrichment function
enrich<-function(list1,list2,pathway){
  l1<-length(list1)
  l2<-length(list2)
  overlap<-length(intersect(list1,list2))
  q<-overlap-1
  m<-l1
  n<-ngenes-l1
  k<-l2
  p<-phyper(q,m,n,k,lower.tail=FALSE,log.p=FALSE)
  return(p)
}

# Fisher P integration function
fisherp<-function (ps) {
  Xsq <- -2 * sum(log(ps))
```

```
  p.val <- pchisq(Xsq, df = 2 * length(ps), lower.tail = FALSE)
  return(p.val)
}

# Regulon vs. pathway
networkUP<-names(regulon[[s2e(tf)]]$tfmode)[regulon[[s2e(tf)]]$tfmode>=0]
networkDN<-names(regulon[[s2e(tf)]]$tfmode)[regulon[[s2e(tf)]]$tfmode<0]

### Enrichment Loop C2
fname<-paste0("results/004_results_hypergeom_C2_",tf,".rda")
#pathways<-msigDBentrez$c2.cp.biocarta.v5.0.entrez.gmt
pathways<-msigDBentrez$c2.all.v5.0.entrez.gmt
if(!file.exists(fname)){
  results<-list()
  ntests<-0
  tfs<-names(regulon)
  pb<-txtProgressBar(0,length(tfs),style=3)
  pbi<-1
  for(tf in tfs){
    plimit<-0.05
    setTxtProgressBar(pb,pbi)
    sublistUP<-setNames(rep(NA,length(pathways)),names(pathways))
    sublistDN<-setNames(rep(NA,length(pathways)),names(pathways))
    i<-1
    for(pathway in pathways){
      networkUP<-names(regulon[[tf]]$tfmode)[regulon[[tf]]$tfmode>=0]
      networkDN<-names(regulon[[tf]]$tfmode)[regulon[[tf]]$tfmode<0]
      pUP<-enrich(networkUP,pathway,ngenes)
      pDN<-enrich(networkDN,pathway,ngenes)
      ntests<-ntests+1
      sublistUP[i]<-pUP
      sublistDN[i]<-pDN
      i<-i+1
    }
    sublistUP<-sublistUP[sublistUP<=plimit]
    sublistDN<-sublistDN[sublistDN<=plimit]
    results[[tf]]<-list(up=sublistUP,dn=sublistDN)
    pbi<-pbi+1
  }
  save(results,ntests,file=fname)
} else {
  load(fname)
}

### Now, plotting
# Set parameters
topn<-20
subsetting<-"REACTOME_"

## Start plot
# Prepare input results
res<-results[[s2e(tf)]]
```

```
# Correct pvalues and integrate
resUP<-p.adjust(res$up,method="none",n=ntests) # convert to bonferroni when deploying
resDN<-p.adjust(res$dn,method="none",n=ntests)
union<-union(names(resUP),names(resDN))
tmptable<-cbind(resUP[union],resDN[union])
tmptable[is.na(tmptable)]<-1
rownames(tmptable)<-union
integrated<-apply(tmptable,1,fisherp)

# Optional subsetting
toshow<-names(sort(integrated))
toshow<-toshow[grep(subsetting,toshow)]
toshow<-toshow[1:topn]


# Prepare input
resUP<--log10(res$up[toshow])
resDN<--log10(res$dn[toshow])
names(resUP)<-toshow
names(resDN)<-toshow
resUP[is.na(resUP)]<-0
resDN[is.na(resDN)]<-0
```

```
# Canvas
upperlim<-ceiling(max(c(resUP,resDN)))
par(mar=c(0,0,0,0))
plot(0,col="white",xlim=c(-10,12),ylim=c(-2,topn+3),xaxt="n",yaxt="n",type="n",frame.plot=FALSE,
     xlab="",ylab="",xaxs="i",yaxs="i")
# text(c(0:10),c(0:10),labels=c(0:10),pos=1,offset=0)
# text(c(1,2,2,1),c(1,2,1,2),labels=c("1_1","2_2","2_1","1_2"),pos=1,offset=0)

# Title
regsize<-length(regulon[[s2e(tf)]]$tfmode)
text(5,topn+2.8,paste0("Pathway enrichment of ",tf," regulon"),pos=1,offset=0,font=2)
text(5,topn+2,paste0("regulon size: ",regsize),pos=1,offset=0,font=3,cex=0.8)

# Axis horizontal
abline(h=0)
segments(0:10,0,0:10,-0.3)
text(5,-1,"Hypergeometric -log10(pvalue)",pos=1,offset=0)
text(1:10,-0.5,cex=0.7,
     labels=round(signif(seq(0,upperlim,length.out=11)),2)[2:11]
)

# Axis vertical
abline(v=0)
#segments(0,topn:0,-0.3,topn:0)

# Significance line
abline(v=(-log10(0.05)/upperlim)*10,lty=2)

# Bars
for(i in topn:1){
  upval<-(resUP[i]/upperlim)*10
```

```
  dnval<-(resDN[i]/upperlim)*10
  rect(0,i+0.25,upval,i,col="salmon")
  rect(0,i,dnval,i-0.25,col="cornflowerblue")
}


# Pathway labels
pathlabels<-gsub(subsetting,"",toshow)
pathlabels<-tolower(pathlabels)
pathlabels<-gsub("_"," ",pathlabels)
for(i in topn:1){
  text(0,i,pathlabels[i],pos=2,offset=0.1,cex=0.8)
}


# Percentage of pathway overlap
r<-names(regulon[[s2e(tf)]]$tfmode)
percs<-c()
for(i in topn:1){
  p<-pathways[[toshow[i]]]
  intrsct<-intersect(p,r)
  perc<-100*length(intrsct)/length(p)
  percs<-c(percs,perc)
}
names(percs)<-rev(toshow)
spercs<-(percs/max(percs))*4
lines(spercs,topn:1,lty=1,lwd=2)

# Ruler of pathway overlap
segments(0,topn+1,4,topn+1)
text(2,topn+1.5,pos=1,offset=0,font=3,cex=0.7,labels="% pathway")
segments(0:4,topn+1,0:4,topn+0.8)
text(1:4,topn+0.65,cex=0.5,
     labels=round(seq(0,max(percs),length.out=5),1)[2:5]
)
```

```
par<-par.backup
```

Comparing VULCAN results with other tools and approaches

Our VULCAN results were obtained using independent gene regulatory models and multiple replicates to generate ER-response signatures.

VULCAN and QRIME

We set to testing the performance of VULCAN against a complementary experimental approach called QRIME (Holding et al., submitted), which aims at identifyin interactors of ER within the ER-chromatin complex at estrogen-responding promoters. We have QRIME results for MCF7 cells treated with estradiol at both 45 and 90 minutes, the same experimental setup for the VULCAN input dataset.

```
# Load imported vulcan object
load("results/001_vobj.rda")
```

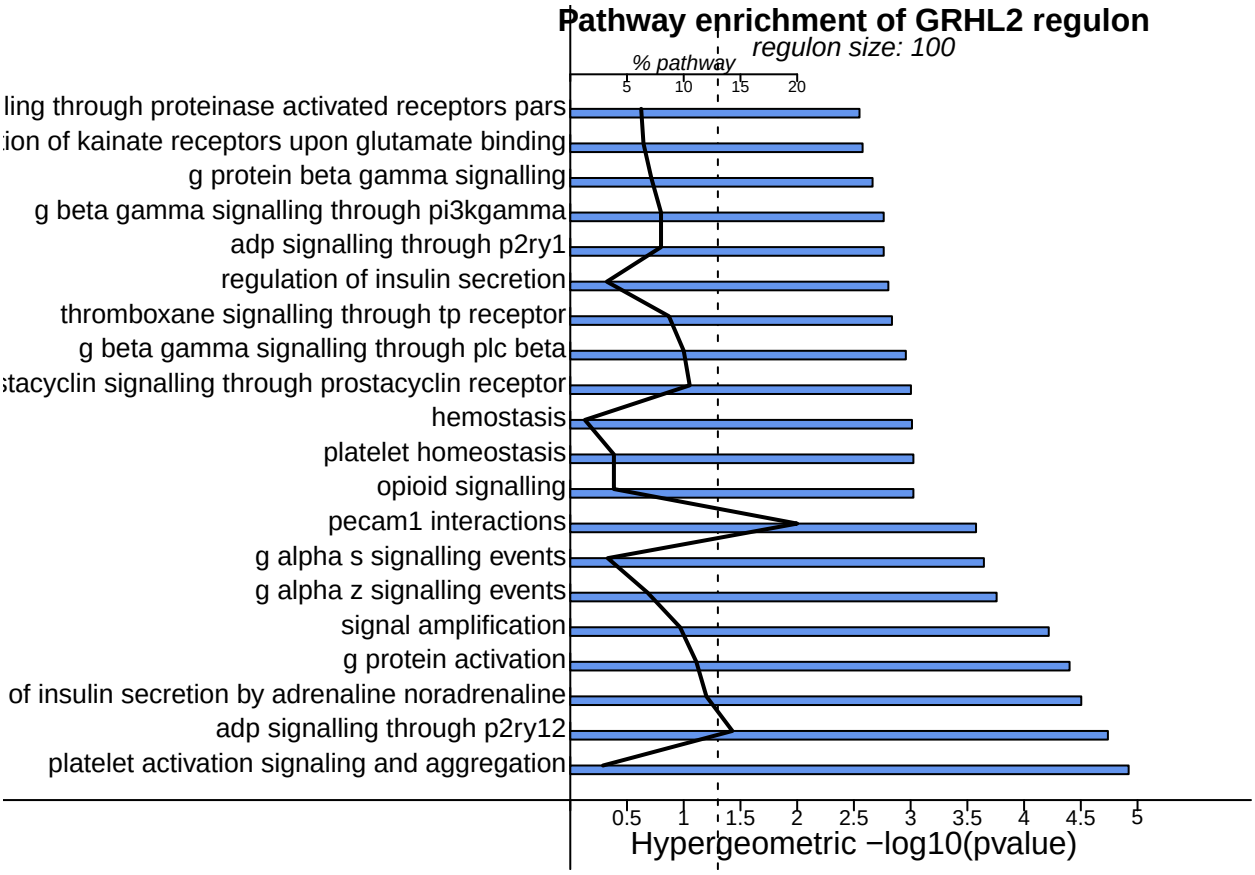


Figure 29: Figure S25. Pathways most significantly overlapping the GRHL2 network

```
#### Vulcan Analysis (multiple networks)
## TCGA network, ARACNe with proteomics centroids
load("aracne/regulon-tcga-centroids.rda")
tcga_regulon<-regulon
rm(regulon)

## METABRIC network, ARACNe with proteomics centroids
load("aracne/regulon-metabric-centroids.rda")
metabric_regulon<-regulon
rm(regulon)

## Specify contrast and network
fname<-"results/005_vobj_networks.rda"
list_eg2symbol<-as.list(org.Hs.egSYMBOL[mappedkeys(org.Hs.egSYMBOL)])
if(!file.exists(fname)){
  vobj_tcga_90<-vulcan(vobj,network=tcga_regulon,contrast=c("t90","t0"),annotation=list_eg2symbol)
  vobj_tcga_45<-vulcan(vobj,network=tcga_regulon,contrast=c("t45","t0"),annotation=list_eg2symbol)
  vobj_metabric_90<-vulcan(vobj,network=metabric_regulon,contrast=c("t90","t0"),annotation=list_eg2symbo
  vobj_metabric_45<-vulcan(vobj,network=metabric_regulon,contrast=c("t45","t0"),annotation=list_eg2symbo
  save(
    vobj_tcga_90,
    vobj_tcga_45,
    vobj_metabric_90,
    vobj_metabric_45,
    file=fname
  )
} else {
  load(fname)
}

# Merge proteomics and TF aracne results
vobj2_tcga_90<-vobj_tcga_90
vobj2_tcga_45<-vobj_tcga_45
vobj2_metabric_90<-vobj_metabric_90
vobj2_metabric_45<-vobj_metabric_45

fname<-"results/002_vobj_networks.rda"
load(fname)
vobj_tcga_90$mrs<-rbind(vobj_tcga_90$mrs,vobj2_tcga_90$mrs[setdiff(rownames(vobj2_tcga_90),rownames(vob
vobj_tcga_45$mrs<-rbind(vobj_tcga_45$mrs,vobj2_tcga_45$mrs[setdiff(rownames(vobj2_tcga_45),rownames(vob
vobj_metabric_90$mrs<-rbind(vobj_metabric_90$mrs,vobj2_metabric_90$mrs[setdiff(rownames(vobj2_metabric_
vobj_metabric_45$mrs<-rbind(vobj_metabric_45$mrs,vobj2_metabric_45$mrs[setdiff(rownames(vobj2_metabric_

# Metabric Results
metabric_90=vobj_metabric_90$mrs[,"pvalue"]
metabric_45=vobj_metabric_45$mrs[,"pvalue"]
tfs<-names(metabric_45)
metabricmat<-cbind(metabric_45[tfs],metabric_90[tfs])
colnames(metabricmat)<-c("T45","T90")

# TCGA Results
tcga_90=vobj_tcga_90$mrs[,"pvalue"]
tcga_45=vobj_tcga_45$mrs[,"pvalue"]
```

```
tfs<-names(tcga_45)
tcgamat<-cbind(tcga_45[tfs],tcga_90[tfs])
colnames(tcgamat)<-c("T45","T90")


## QRIME results
raw45<-read.delim("qrime/Results/ER_45min_vs_ER_0minExcludeMissingValues_QuantileNormalization.txt",as.
qrime45<-setNames(raw45$P.Value,raw45$Gene)
qrime45<-qrime45[!is.na(qrime45)]

raw90<-read.delim("qrime/Results/ER_90min_vs_ER_0minExcludeMissingValues_QuantileNormalization.txt",as.
qrime90<-setNames(raw90$P.Value,raw90$Gene)
qrime90<-qrime90[!is.na(qrime90)]
```

Here, we will compare ER-binding pvalues obtained with QRIME, with the VULCAN results aimed at identifying TFs upstream of the observed differential promoter binding.

```
### Comparisons, pvalues
par(mfrow=c(2,2))
# TCGA 45
main<-"TCGA network, 45 mins"
common<-intersect(rownames(tcgamat),names(qrime45))
x<-qrime45[common]
x<--log10(x)
y<-tcgamat[common,"T45"]
y<--log10(y)
plot(x,y,pch=20,xlab="-log10(p) QRIME",ylab="-log10(p) VULCAN",col="grey",main=main,
     xlim=c(min(x)-max(x)*0.3,max(x)*1.1),
     ylim=c(min(y)-max(y)*0.2,max(y)*1.1)
)
topx<-names(sort(x,dec=TRUE))[1:10]
topy<-names(sort(y,dec=TRUE))[1:10]
top<-union(topx,topy)
textplot2(x[top],y[top],words=top,new=FALSE)
grid()

# TCGA 90
main<-"TCGA network, 90 mins"
common<-intersect(rownames(tcgamat),names(qrime90))
x<-qrime90[common]
x<--log10(x)
y<-tcgamat[common,"T90"]
y<--log10(y)
plot(x,y,pch=20,xlab="-log10(p) QRIME",ylab="-log10(p) VULCAN",col="grey",main=main,
     xlim=c(min(x)-max(x)*0.3,max(x)*1.1),
     ylim=c(min(y)-max(y)*0.2,max(y)*1.1)
)
topx<-names(sort(x,dec=TRUE))[1:10]
topy<-names(sort(y,dec=TRUE))[1:10]
top<-union(topx,topy)
textplot2(x[top],y[top],words=top,new=FALSE)
grid()
```

```
# METABRIC 45
main<-"METABRIC network, 45 mins"
common<-intersect(rownames(metabricmat),names(qrime45))
x<-qrime45[common]
x<--log10(x)
y<-metabricmat[common,"T45"]
y<--log10(y)
plot(x,y,pch=20,xlab="-log10(p) QRIME",ylab="-log10(p) VULCAN",col="grey",main=main,
     xlim=c(min(x)-max(x)*0.3,max(x)*1.1),
     ylim=c(min(y)-max(y)*0.2,max(y)*1.1)
)
topx<-names(sort(x,dec=TRUE))[1:10]
topy<-names(sort(y,dec=TRUE))[1:10]
top<-union(topx,topy)
textplot2(x[top],y[top],words=top,new=FALSE)
grid()

# METABRIC 90
main<-"METABRIC network, 90 mins"
common<-intersect(rownames(metabricmat),names(qrime90))
x<-qrime90[common]
x<--log10(x)
y<-metabricmat[common,"T90"]
y<--log10(y)
plot(x,y,pch=20,xlab="-log10(p) QRIME",ylab="-log10(p) VULCAN",col="grey",main=main,
     xlim=c(min(x)-max(x)*0.3,max(x)*1.1),
     ylim=c(min(y)-max(y)*0.2,max(y)*1.1)
)
topx<-names(sort(x,dec=TRUE))[1:10]
topy<-names(sort(y,dec=TRUE))[1:10]
top<-union(topx,topy)
textplot2(x[top],y[top],words=top,new=FALSE)
grid()
```

```
# TCGA Results
tcganes_90=vobj_tcga_90$mrs[,"NES"]
tcganes_45=vobj_tcga_45$mrs[,"NES"]
tfs<-names(tcganes_45)
tcganesmat<-cbind(tcganes_45[tfs],tcganes_90[tfs])
colnames(tcganesmat)<-c("T45","T90")

# METABRIC Results
metabricnes_90=vobj_metabric_90$mrs[,"NES"]
metabricnes_45=vobj_metabric_45$mrs[,"NES"]
tfs<-names(metabricnes_45)
metabricnesmat<-cbind(metabricnes_45[tfs],metabricnes_90[tfs])
colnames(metabricnesmat)<-c("T45","T90")



par(mfrow=c(2,2))
# TCGA 45
main<-"TCGA network, 45 mins"
```

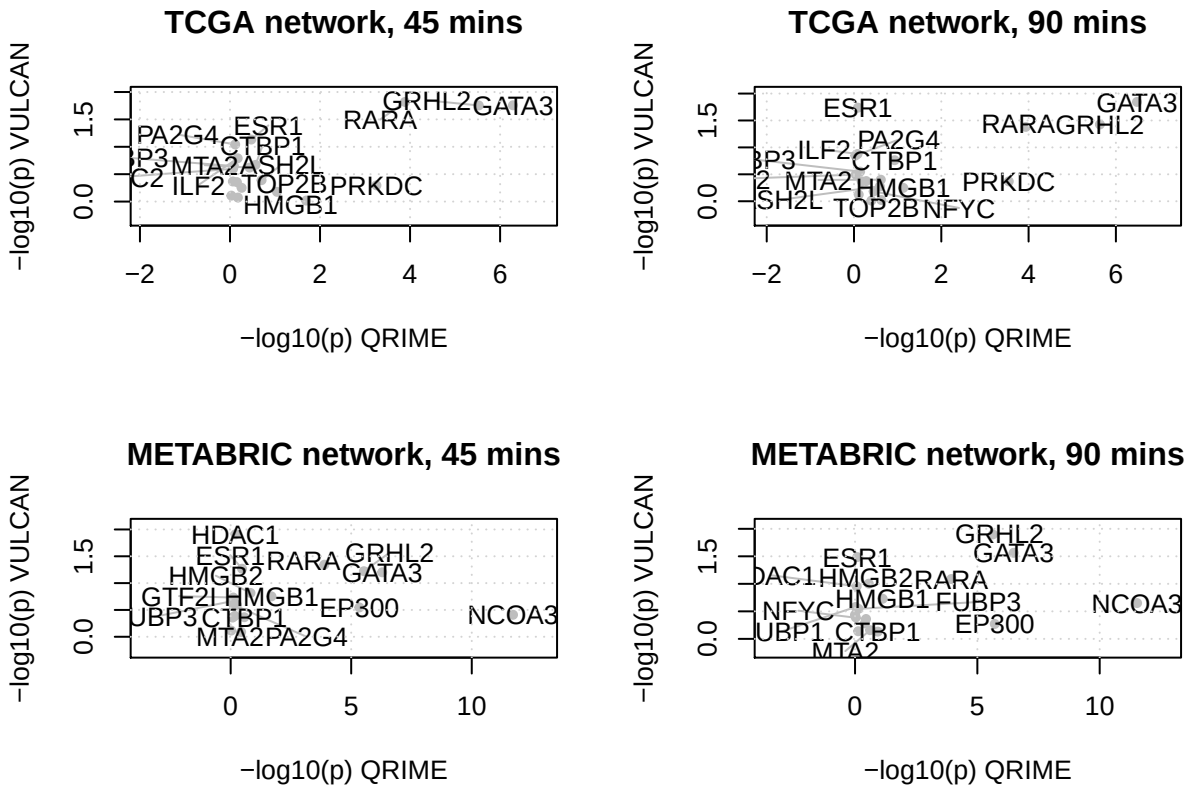


Figure 30: Figure S26. Comparison of significance between the QRIME method (x-axis) and the VULCAN method (y-axis) at two time points using two regulatory networks for VULCAN

```
common<-intersect(rownames(tcganesmat),names(qrime45))
x<-qrime45[common]
x<--log10(x)
y<-tcganesmat[common,"T45"]
plot(x,y,pch=20,xlab="-log10(p) QRIME",ylab="NES VULCAN",col="grey",main=main,
     xlim=c(min(x)-max(x)*0.3,max(x)*1.1),
     ylim=c(min(y)-max(y)*0.2,max(y)*1.1)
)
topx<-names(sort(x,dec=TRUE))[1:10]
topy<-names(sort(y,dec=TRUE))[1:10]
top<-union(topx,topy)
textplot2(x[top],y[top],words=top,new=FALSE)
abline(h=c(-p2z(0.05),p2z(0.05)),lty=3)
grid()

# TCGA 90
main<-"TCGA network, 90 mins"
common<-intersect(rownames(tcganesmat),names(qrime90))
x<-qrime90[common]
x<--log10(x)
y<-tcganesmat[common,"T90"]
plot(x,y,pch=20,xlab="-log10(p) QRIME",ylab="NES VULCAN",col="grey",main=main,
     xlim=c(min(x)-max(x)*0.3,max(x)*1.1),
     ylim=c(min(y)-max(y)*0.2,max(y)*1.1)
)
topx<-names(sort(x,dec=TRUE))[1:10]
topy<-names(sort(y,dec=TRUE))[1:10]
top<-union(topx,topy)
textplot2(x[top],y[top],words=top,new=FALSE)
abline(h=c(-p2z(0.05),p2z(0.05)),lty=3)
grid()


# METABRIC 45
main<-"METABRIC network, 45 mins"
common<-intersect(rownames(metabricnesmat),names(qrime45))
x<-qrime45[common]
x<--log10(x)
y<-metabricnesmat[common,"T45"]
plot(x,y,pch=20,xlab="-log10(p) QRIME",ylab="NES VULCAN",col="grey",main=main,
     xlim=c(min(x)-max(x)*0.3,max(x)*1.1),
     ylim=c(min(y)-max(y)*0.2,max(y)*1.1)
)
topx<-names(sort(x,dec=TRUE))[1:10]
topy<-names(sort(y,dec=TRUE))[1:10]
top<-union(topx,topy)
textplot2(x[top],y[top],words=top,new=FALSE)
abline(h=c(-p2z(0.05),p2z(0.05)),lty=3)
grid()

# METABRIC 90
main<-"METABRIC network, 90 mins"
common<-intersect(rownames(metabricnesmat),names(qrime90))
```

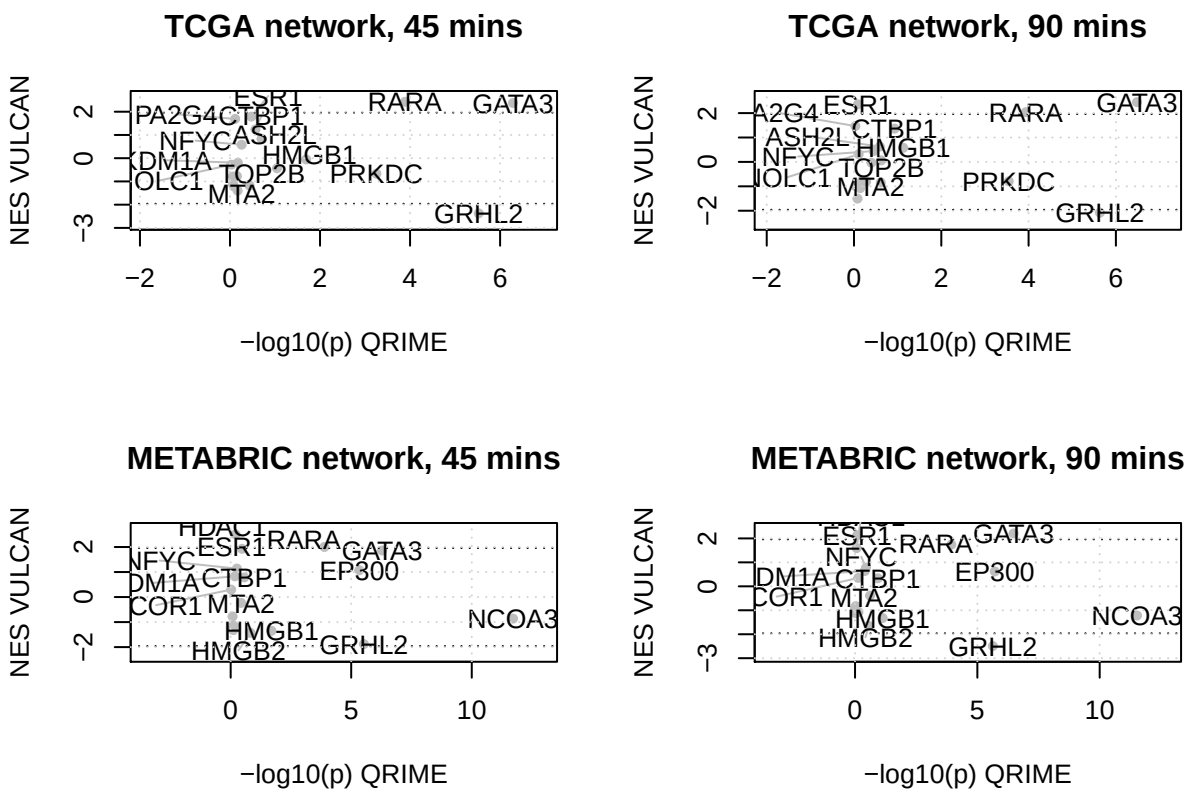


Figure 31: Figure S27. Comparison of Normalized Enrichment Score between the QRIME method (x-axis) and the VULCAN method (y-axis) at two time points using two regulatory networks for VULCAN

```
x<-qrime90[common]
x<--log10(x)
y<-metabricnesmat[common,"T90"]
plot(x,y,pch=20,xlab="-log10(p) QRIME",ylab="NES VULCAN",col="grey",main=main,
     xlim=c(min(x)-max(x)*0.3,max(x)*1.1),
     ylim=c(min(y)-max(y)*0.2,max(y)*1.1)
)
topx<-names(sort(x,dec=TRUE))[1:10]
topy<-names(sort(y,dec=TRUE))[1:10]
top<-union(topx,topy)
textplot2(x[top],y[top],words=top,new=FALSE)
abline(h=c(-p2z(0.05),p2z(0.05)),lty=3)
grid()
```

Comparing Mutual Information and Partial Correlation networks

As shown before in literature, partial correlation and mutual information networks are highly similar. While requiring a mutual information network due to the VIPER engine used by the VULCAN pipeline, we set out here to compare Mutual Information networks with partial correlation ones, using different correlation thresholds.

```
filename<-"results/006_comparison1_aracne.rda"
if(!file.exists(filename)){
  # Load aracne network
```

```r
load("aracne/regulon-tcga-centroids.rda")

# Load expression matrix
expmat<-as.matrix(read.delim("aracne/tcga-expmat.dat",as.is=TRUE,row.names=1))


### Better pipeline
tfs<-names(regulon)
genes<-rownames(expmat)


# Remove zero-SD genes
sds<-apply(expmat,1,sd)
expmat<-expmat[sds>=0.5,]
dim(expmat)



# Build Covariance Matrix
pcormat<-cov(t(expmat))
# Build pcormat
pcormat<-cor2pcor(pcormat)
rownames(pcormat)<-colnames(pcormat)<-rownames(expmat)


# Keep only TFs
diag(pcormat)<-0
tfs<-intersect(tfs,rownames(pcormat))
pcormat<-pcormat[tfs,]

# Transform my network into a vector of edges
myedges<-c()
for(centroid in names(regulon)){
  targets<-names(regulon[[centroid]]$tfmode)
  newedges<-paste0(centroid,"___",targets)
  myedges<-c(myedges,newedges)
}
length(myedges) # 268394

### Loop over rs
jis<-c()
nedges<-c()
rs<-seq(0.05,0.4,by=0.025)
for(r in rs){
  # Transform networks into vectors of edges
  matches<-which(abs(pcormat)>=r,arr.ind=TRUE)
  pcoredges<-paste0(rownames(pcormat)[matches[,1]],"___",colnames(pcormat)[matches[,2]])
  nedges<-c(nedges,nrow(matches))
  # Calculate Jaccard
  ji<-length(intersect(myedges,pcoredges))/length(union(myedges,pcoredges))
  jis<-c(jis,ji)
}
names(jis)<-names(nedges)<-rs
```

```
  # Generate random network vectors
  random<-list()
  for(ii in 1:length(rs)){
    n<-nedges[ii]
    message("Doing ",rs[ii])
    pb<-txtProgressBar(0,1000,style=3)
    randomjis<-c()
    for(i in 1:1000){
      x<-sample(1:nrow(pcormat),n,replace=TRUE)
      y<-sample(1:ncol(pcormat),n,replace=TRUE)
      matches<-cbind(x,y)
      pcoredges<-paste0(rownames(pcormat)[matches[,1]],"___",colnames(pcormat)[matches[,2]])
      ji<-length(intersect(myedges,pcoredges))/length(union(myedges,pcoredges))
      randomjis<-c(randomjis,ji)
      setTxtProgressBar(pb,i)
    }
    random[[ii]]<-randomjis
  }
  names(random)<-as.character(rs)

  save(jis,rs,nedges,random,file=filename)

} else {
  load(filename)
}
```

We generate several partial correlation networks using the same input as the ARACNe network used by VULCAN (the TCGA breast cancer dataset). We tested the overlap of every partial correlation network with the ARACNe network using the Jaccard Index (JI) criterion.

```
par(mfrow=c(1,1))
bp<-barplot(jis,xlab="pcor r",ylab="JI",ylim=c(0,max(jis)*1.1))
text(bp[,1],jis,labels=kmgformat(nedges),pos=3,cex=0.7)
mtext("Number of overlapping edges between Pcor and Aracne",cex=0.8)
```

Finally, we show how the Jaccard Index between partial correlation networks and the ARACNe network is always significantly higher than expected by selecting random network edges.

```
par(mfrow=c(3,5))
for(i in 1:length(random)){
  dd<-density(random[[i]])
  plot(dd,xlim=c(min(random[[i]]),jis[i]*1.1),main=paste0("r=",rs[i]),xlab="Jaccard Index",lwd=2)
  arrows(jis[i],max(dd$y),jis[i],0,lwd=2)
}
```

Comparing alternative methods for target enrichment analysis

We developed three independent methods to compare VULCAN with. The first is common, the second is simple but crude, and the third is also common but too stringent.

```
### Classic vulcan result
# vobj_tcga_90<-vulcan(vobj,network=tcga_regulon,contrast=c("t90","t0"),annotation=list_eg2symbol)
```

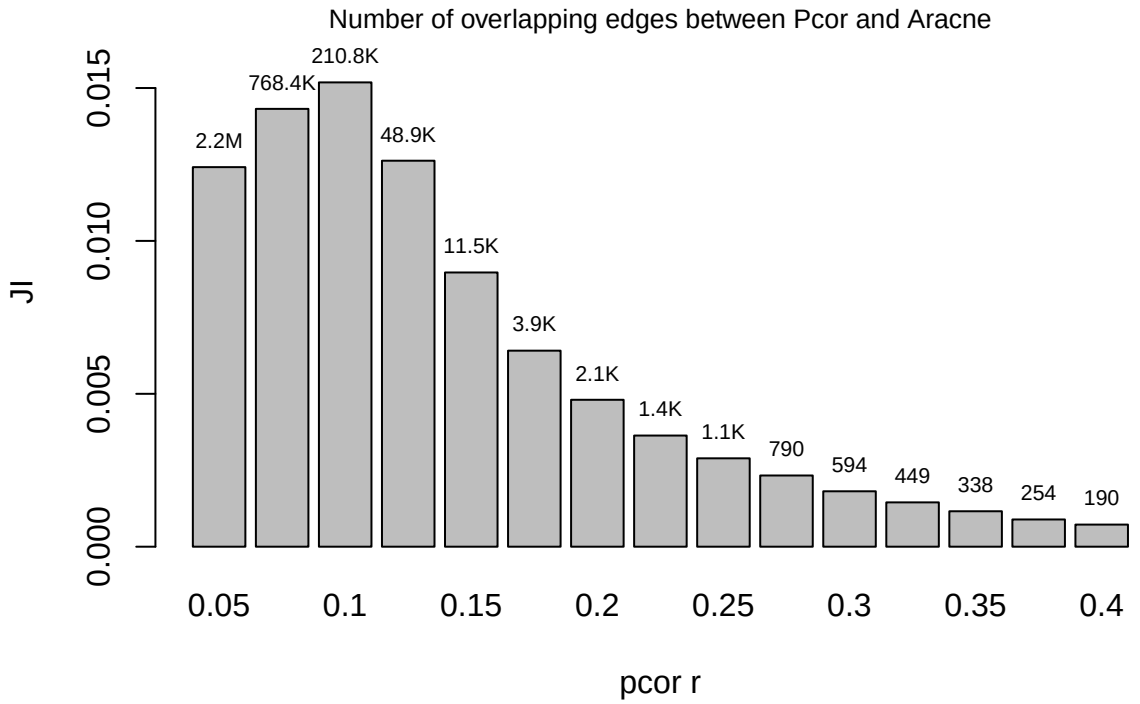


Figure 32: Figure S28. Sheer number of overlapping edges at different Pcor thresholds

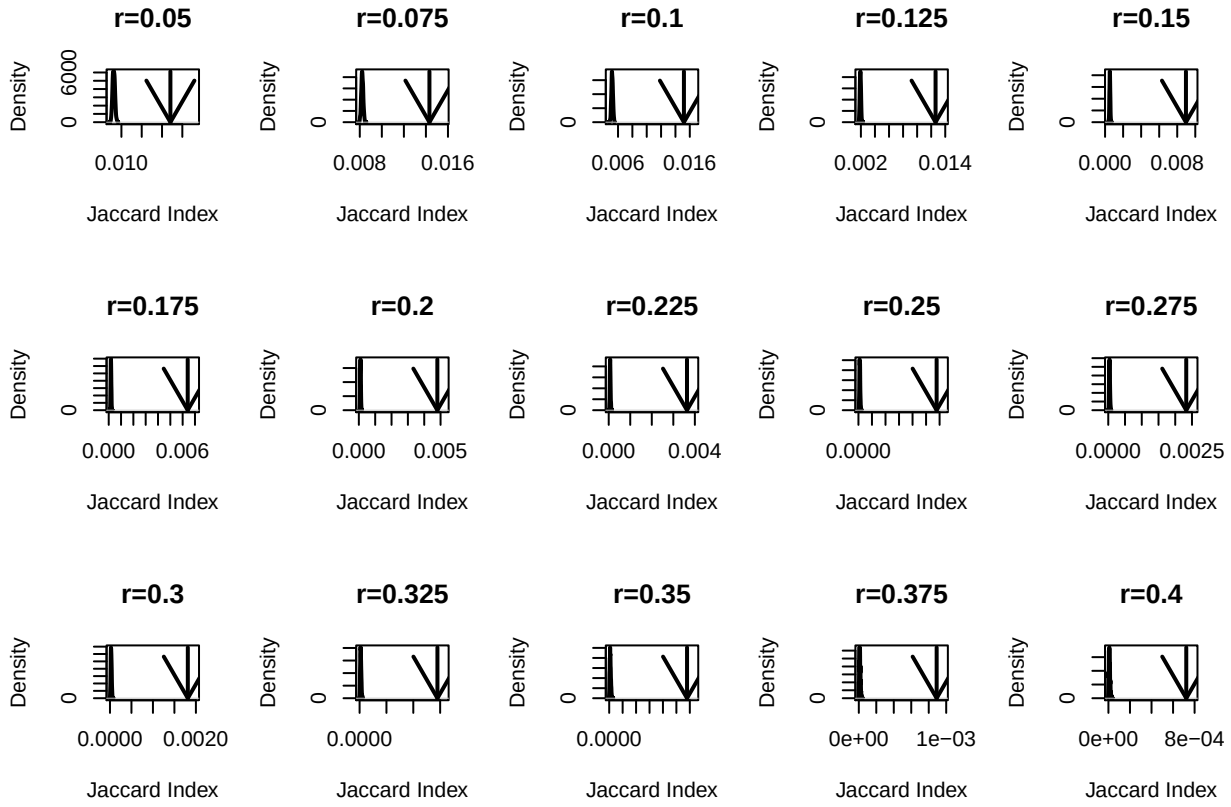


Figure 33: Figure S29. Comparing the Jaccard Index between the ARACNe network and Partial correlation networks (indicated as arrows) and random networks (indicated as distributions)

```
wass<-load("results/002_vobj_networks.rda")

### Prepare the workspace
# Load imported vulcan object
load("results/001_vobj.rda")

# Network
load("networks/brca-tf-regulon.rda")
tcga_regulon<-regul
rm(regul)

# Entrez to symbol
library(org.Hs.eg.db)
list_eg2symbol<-as.list(org.Hs.egSYMBOL[mappedkeys(org.Hs.egSYMBOL)])
```

1. A T-test based method, that takes the targets of a TF and integrates their pvalue in a specific contrast. Good but hard to keep control over the p-value (the classic integration step using the Fisher's Integration method vastly underestimates them). Also, GRHL2 doesn't shine with this method. Unless differently specified, the p-values are always Bonferroni-corrected

```
############################################################
### TTEST INTEGRATION
############################################################
if(TRUE){
  ### TTest functions
  msvipertt<-function(signature,network,minsize=10){
    # First, convert tscores into pvalues
    psignature<-2*pt(signature[,1], nrow(signature)-2, lower=FALSE)

    # For each regulon, take the targets and integrate their pvalue
    tfpvalues<-c()
    for(tf in names(network)){
      targets<-names(network[[tf]]$tfmode)
      ptargets<-psignature[intersect(targets,names(psignature))]
      if(length(ptargets)>=minsize){
        pt<-fisherp(ptargets)
        tfpvalues<-c(tfpvalues,pt)
        names(tfpvalues)[length(tfpvalues)]<-tf
      }
    }

    return(tfpvalues)
  }
  vulcantt<-function(vobj, network, contrast, annotation = NULL, minsize = 10) {
    tfs <- names(network)
    samples <- vobj$samples
    normalized <- vobj$normalized

    # Prepare output objects
    msvipers <- matrix(NA, ncol = 3, nrow = length(tfs))
    rownames(msvipers) <- tfs
    # Define contrast
    a <- samples[[contrast[1]]]
    b <- samples[[contrast[2]]]
```

```
    # Vulcan msviper implementation
    signature <- rowTtest(normalized[, a], normalized[, b])$statistic
    tfpvalues <- msvipertt(signature, network, minsize = minsize)
    if(!is.null(annotation)){
      names(tfpvalues)<-annotation[names(tfpvalues)]
    }
    return(tfpvalues)
  }

  ### Repeat the same analysis but with TT
  ttp_90<-p.adjust(vulcantt(vobj,network=tcga_regulon,contrast=c("t90","t0"),annotation=list_eg2symbol)
  ttp_45<-p.adjust(vulcantt(vobj,network=tcga_regulon,contrast=c("t45","t0"),annotation=list_eg2symbol)

  ### Compare vulcan and TT
  vulcanp_90<-vobj_tcga_90$msviper$es$p.value
  vulcanp_45<-vobj_tcga_45$msviper$es$p.value

  toshow<-c("ESR1","GATA3","GRHL2")

  par(mfrow=c(1,2))

  common<-intersect(names(ttp_45),names(vulcanp_45))
  x<--log10(vulcanp_45[common]+1e-320)
  y<--log10(ttp_45[common]+1e-320)
  plot(x,y,pch=20,main="Method Comparison, 45' vs 00'",xlab="VULCAN pvalue",ylab="Ttest integration pval
  grid()
  pcc<-cor.test(x,y)
  mtext(paste0("PCC=",signif(pcc$estimate,3)," (p=",signif(pcc$p.value,3),")"))
  textplot2(x[toshow],y[toshow],words=toshow,new=FALSE)

  common<-intersect(names(ttp_90),names(vulcanp_90))
  x<--log10(vulcanp_90[common]+1e-320)
  y<--log10(ttp_90[common]+1e-320)
  plot(x,y,pch=20,main="Method Comparison, 90' vs 00'",xlab="VULCAN pvalue",ylab="Ttest integration pval
  grid()
  pcc<-cor.test(x,y)
  mtext(paste0("PCC=",signif(pcc$estimate,3)," (p=",signif(pcc$p.value,3),")"))
  textplot2(x[toshow],y[toshow],words=toshow,new=FALSE)

  par(mfrow=c(1,1))

}
```

2. A fraction of targets method, defining for every TF the fraction of their targets that are also differentially bound. A crude alternative to VULCAN, which is ignoring the MI strength of interaction and the individual strengths of differential bindings.

```
############################################################
### FRACTION TEST
############################################################
if(TRUE){
  ### TTest functions
  msviperfrac<-function(signature,network,minsize=10){
    # First, convert tscores into pvalues
```

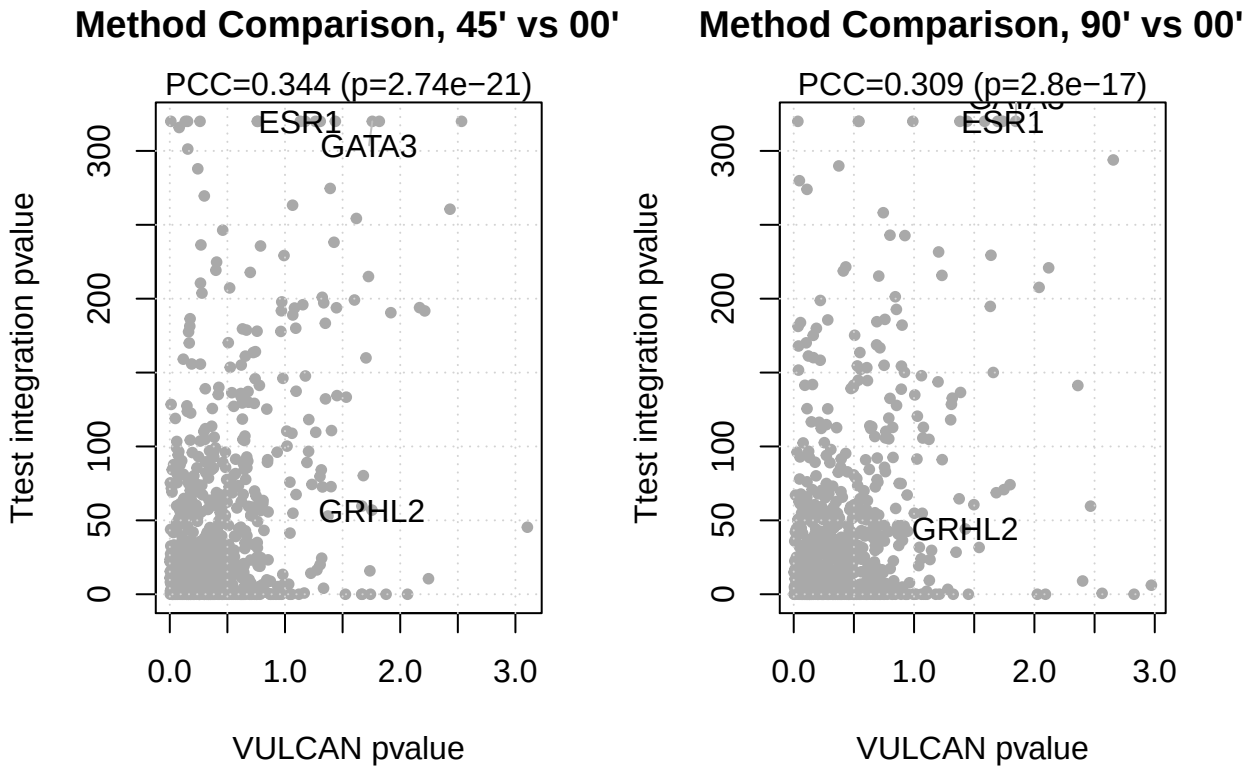


Figure 34: Figure S30. Comparison between VULCAN/VIPER and T-test integration

```
  psignature<-2*pt(signature[,1], nrow(signature)-2, lower=FALSE)

  # For each regulon, take the targets and integrate their pvalue
  tfpvalues<-c()
  for(tf in names(network)){
    targets<-names(network[[tf]]$tfmode)
    ptargets<-psignature[intersect(targets,names(psignature))]
    ptargets<-ptargets
    frac<-length(ptargets)/length(targets)
    tfpvalues<-c(tfpvalues,frac)
    names(tfpvalues)[length(tfpvalues)]<-tf
  }
  return(tfpvalues)
}
vulcanfrac<-function(vobj, network, contrast, annotation = NULL, minsize = 10) {
  tfs <- names(network)
  samples <- vobj$samples
  normalized <- vobj$normalized

  # Prepare output objects
  msvipers <- matrix(NA, ncol = 3, nrow = length(tfs))
  rownames(msvipers) <- tfs
  # Define contrast
  a <- samples[[contrast[1]]]
  b <- samples[[contrast[2]]]
  # Vulcan msviper implementation
```

```
    signature <- rowTtest(normalized[, a], normalized[, b])$statistic
    tfpvalues <- msviperfrac(signature, network, minsize = minsize)
    if(!is.null(annotation)){
      names(tfpvalues)<-annotation[names(tfpvalues)]
    }
    return(tfpvalues)
  }

  ### Repeat the same analysis but with TT
  ttp_90<-vulcanfrac(vobj,network=tcga_regulon,contrast=c("t90","t0"),annotation=list_eg2symbol)
  ttp_45<-vulcanfrac(vobj,network=tcga_regulon,contrast=c("t45","t0"),annotation=list_eg2symbol)

  ### Compare vulcan and TT
  vulcanp_90<-vobj_tcga_90$msviper$es$p.value
  vulcanp_45<-vobj_tcga_45$msviper$es$p.value

  toshow<-c("ESR1","GATA3","GRHL2")

  par(mfrow=c(1,2))

  common<-intersect(names(ttp_45),names(vulcanp_45))
  x<--log10(vulcanp_45[common])
  y<-ttp_45[common]
  plot(x,y,pch=20,main="Method Comparison, 45' vs 00'",xlab="VULCAN pvalue",ylab="Fraction of Differenti
  grid()
  pcc<-cor.test(x,y)
  mtext(paste0("PCC=",signif(pcc$estimate,3)," (p=",signif(pcc$p.value,3),")"))
  textplot2(x[toshow],y[toshow],words=toshow,new=FALSE)

  common<-intersect(names(ttp_90),names(vulcanp_90))
  x<--log10(vulcanp_90[common])
  y<-ttp_90[common]
  plot(x,y,pch=20,main="Method Comparison, 90' vs 00'",xlab="VULCAN pvalue",ylab="Fraction of Differenti
  grid()
  pcc<-cor.test(x,y)
  mtext(paste0("PCC=",signif(pcc$estimate,3)," (p=",signif(pcc$p.value,3),")"))
  textplot2(x[toshow],y[toshow],words=toshow,new=FALSE)

  par(mfrow=c(1,1))

}
```

3. A Fisher's Exact Method (different from the Fisher's Integration!) which assesses the overlap between networks and significant differential binding. Here the problem is that the test is too stringent (as observed in the original VIPER paper) and even without p-value correction we get nothing significant (I tried different threshold of differential binding significance). The plot is uninspiring highlights the non-significance of the results of this method:

```
#############################################################
### FISHER'S EXACT TEST
#############################################################
if(TRUE){
  ### TTest functions
  msviperfet<-function(signature,network,minsize=10){
```

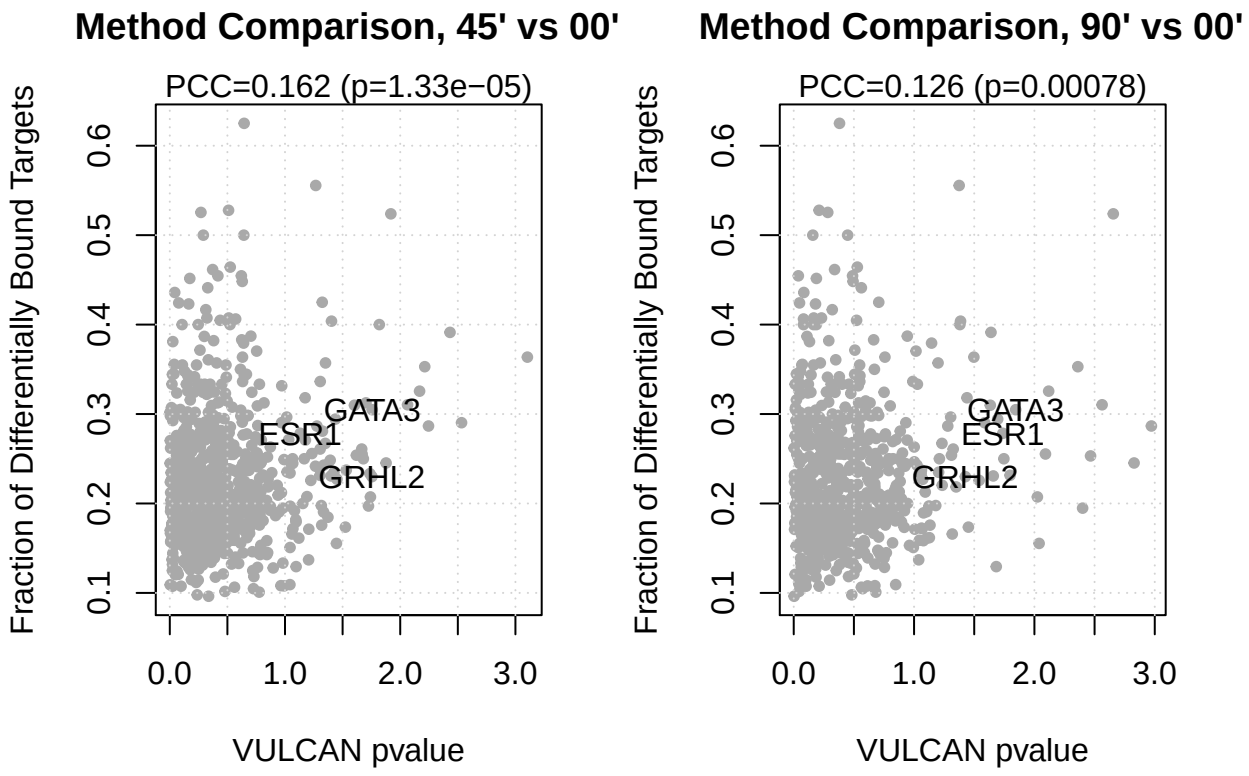


Figure 35: Figure S31. Comparison between VULCAN/VIPER and a fraction of targets found method

```
  # First, convert tscores into pvalues
  psignature<-2*pt(signature[,1], nrow(signature)-2, lower=FALSE)

  # For each regulon, take the targets and calculate how many are significant
  tfpvalues<-c()
  for(tf in names(network)){
    targets<-names(network[[tf]]$tfmode)
    ptargets<-psignature[intersect(targets,names(psignature))]
    if(length(ptargets)>=minsize){
      # Prepare the contingency table
      ul<-names(ptargets)[ptargets<=0.05]
      ur<-setdiff(targets,ul)
      dl<-setdiff(names(psignature)[psignature<=0.05],ul)
      ctable<-rbind(
        c(length(ul),length(ur)),
        c(length(dl),0)
      )
      #ctable[2,2]<-5000-sum(ctable)
      fet<-fisher.test(ctable,alternative="greater")
      pt<-fet$p.value
      tfpvalues<-c(tfpvalues,pt)
      names(tfpvalues)[length(tfpvalues)]<-tf
    }
  }

  return(tfpvalues)
```

```
  }
  vulcanfet<-function(vobj, network, contrast, annotation = NULL, minsize = 10) {
    tfs <- names(network)
    samples <- vobj$samples
    normalized <- vobj$normalized

    # Prepare output objects
    msvipers <- matrix(NA, ncol = 3, nrow = length(tfs))
    rownames(msvipers) <- tfs
    # Define contrast
    a <- samples[[contrast[1]]]
    b <- samples[[contrast[2]]]
    # Vulcan msviper implementation
    signature <- rowTtest(normalized[, a], normalized[, b])$statistic
    tfpvalues <- msviperfet(signature, network, minsize = minsize)
    if(!is.null(annotation)){
      names(tfpvalues)<-annotation[names(tfpvalues)]
    }
    return(tfpvalues)
  }

  ### Repeat the same analysis but with FET
  ttp_90<-vulcanfet(vobj,network=tcga_regulon,contrast=c("t90","t0"),annotation=list_eg2symbol)
  ttp_45<-vulcanfet(vobj,network=tcga_regulon,contrast=c("t45","t0"),annotation=list_eg2symbol)

  ### Compare vulcan and TT
  vulcanp_90<-vobj_tcga_90$msviper$es$p.value
  vulcanp_45<-vobj_tcga_45$msviper$es$p.value

  toshow<-c("ESR1","GATA3","GRHL2")

  par(mfrow=c(1,2))

  common<-intersect(names(ttp_45),names(vulcanp_45))
  x<--log10(vulcanp_45[common]+1e-320)
  y<--log10(ttp_45[common]+1e-320)
  plot(x,y,pch=20,main="Method Comparison, 45' vs 00'",xlab="VULCAN pvalue",ylab="Classic Fisher's Exac
  grid()
  textplot2(x[toshow],y[toshow],words=toshow,new=FALSE)

  common<-intersect(names(ttp_90),names(vulcanp_90))
  x<--log10(vulcanp_90[common]+1e-320)
  y<--log10(ttp_90[common]+1e-320)
  plot(x,y,pch=20,main="Method Comparison, 90' vs 00'",xlab="VULCAN pvalue",ylab="Classic Fisher's Exac
  grid()
  textplot2(x[toshow],y[toshow],words=toshow,new=FALSE)

  par(mfrow=c(1,1))
}
```

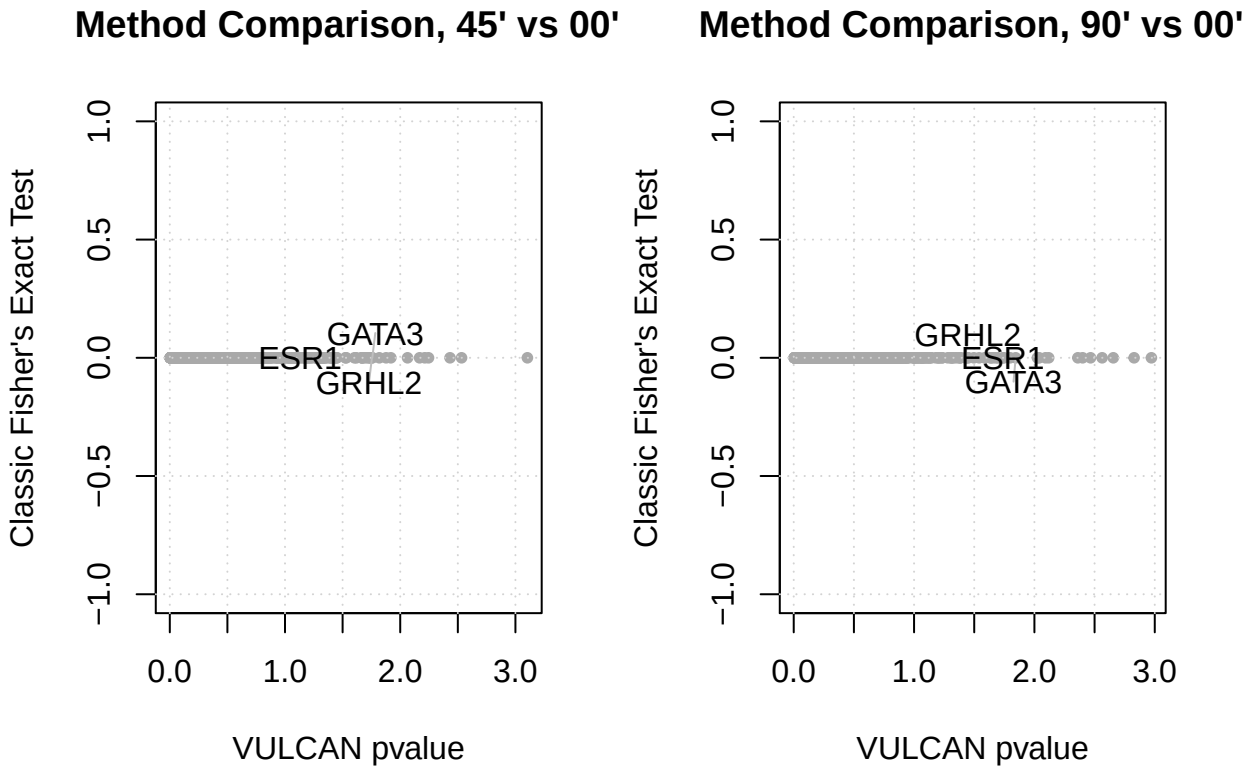


Figure 36: Figure S32. Comparison between VULCAN/VIPER and Fisher's Exact Test method

Comparing VULCAN with online tools

We will generate BED files for the peaks, as reported by DiffBind, we will then test the pathway enrichment for each of these peaks using the Great software v3.0.0 (http://bejerano.stanford.edu/great/public/html/), the ChIP-Enrich pipeline (http://chip-enrich.med.umich.edu/) and the ISMARA tool (http://ismara.unibas.ch). Parameters for GREAT: defaults (Basal plus extension, 5.0kb upstream, 1.0 kb downstream) Parameters for ChIP-Enrich: defaults (Nearest TSS, pathways size<=2000, Biocarta, KEGG, Reactome, TFs). Parameters for ISMARA: defaults (ChIP-Seq mode, hg19)

The VULCAN analysis shows a significant overlap in terms of significant pathways with the GREAT method. ChIP-enrich computes enrichment for a number of TFs which are amongst the most significant in VULCAN, but it surprisingly fails at identifing ESR1 as the top Transcription Factor affected by our experiment. ISMARA succeeds at identifying ESR1 using a motif-based analysis, but doesn't identify other candidate binding TFs, as expected, being the experiment targeted at the estrogen receptor.

```
### Number of tested pathways (for statistical comparison later)
load("msigdb/MSigDB_v5.0_human.rda")
raw<-msigDBentrez$c2.all.v5.0.entrez.gmt
universe<-names(raw)[grep("BIOCARTA_|REACTOME_|KEGG_|PID_|ST_",names(raw))]

### DiffBind object (containing peak location and intensity)
wass<-load("results/001_diffbind.rda")

### Generate input BEDs for chip enrich and GREAT (ismara takes BAMs as input)
fname<-paste0("results/008_comparison_contrast90vs00_UP_p1.bed")
if(!file.exists(fname)){
  contrasts<-c("contrast45vs00","contrast90vs00","contrast90vs45")
```

```
  for (contrast in contrasts) {
    c<-switch(contrast,contrast45vs00=3,contrast90vs00=2,contrast90vs45=1)
    for(pgreat in c(1,0.05)){
      bed<-as.data.frame(dba.report(dbaobj,contrast=c,method=DBA_DESEQ2,bNormalized=TRUE, bCounts=TRUE,
      if(nrow(bed)>0) { # Nothing individually significant if no rows were produced
        bedup<-bed[bed$Fold>0,1:3]
        beddn<-bed[bed$Fold<0,1:3]
        if(nrow(bedup)>0){
          write.table(bedup,
                      file=paste0("results/008_comparison_",contrast,"_UP_p",pgreat,".bed"),
                      row.names=FALSE,col.names=FALSE,sep="\t",quote=FALSE
          )
        }
        if(nrow(beddn)>0){
          write.table(beddn,
                      file=paste0("results/008_comparison_",contrast,"_DN_p",pgreat,".bed"),
                      row.names=FALSE,col.names=FALSE,sep="\t",quote=FALSE
          )
        }
      }
    }
  }
}
```

```
### Pathway Comparison: VULCAN vs. GREAT
load_obj <- function(f){
  env <- new.env()
  nm <- load(f, env)[1]
  env[[nm]]
}
contrasts<-c("45","90")
par(mfrow=c(1,2))
for(c in contrasts){
  # Our REA pathways
  nes.tpathways<-load_obj(paste0("results/003_pathwayComparison_REA_",c,".rda")) # results_rea_45

  # GREAT pathways
  rawgreat<-read.delim(paste0("results/great/great_",c,"_p1_UP.tsv"),as.is=TRUE,skip=3)
  table(rawgreat[,1])
  rawgreat<-rawgreat[rawgreat[,1]=="MSigDB Pathway",]
  great<-setNames(rawgreat$BinomBonfP,rawgreat[,2])
  great<-great[great<0.1]

  ### Comparison GREAT/VULCAN
  vsig<-nes.tpathways[z2p(nes.tpathways)<0.1]
  venn(list(VULCAN=names(vsig),GREAT=names(great)))
  title(paste0("Comparison VULCAN/GREAT for pathways at ",c," minutes"))
  ctable<-rbind(c(0,0),c(0,0))
  ctable[1,1]<-length(intersect(names(vsig),names(great)))
  ctable[1,2]<-length(setdiff(names(vsig),names(great)))
  ctable[2,1]<-length(setdiff(names(great),names(vsig)))
  ctable[2,2]<-length(universe)-ctable[1,1]-ctable[1,2]-ctable[2,1]
  fp<-signif(fisher.test(ctable)$p.value,4)
  mtext(paste0("FET p-value: ",fp))
```

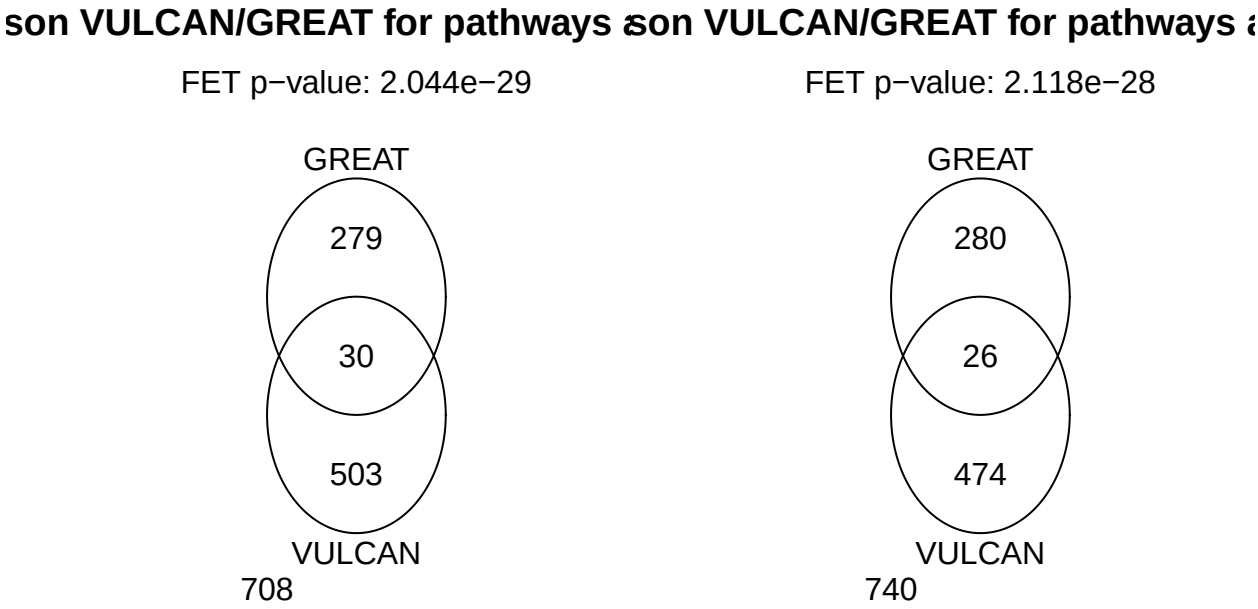


Figure 37: Figure S33. Comparison of results from the VULCAN and GREAT methods

```
  text(100,3,ctable[2,2])

  common<-intersect(names(nes.tpathways),names(great))
}
```

```
par(mfrow=c(1,1))
```

```
### TF and Pathway Comparison: VULCAN vs. ChIP enrich
load("results/002_vobj_networks.rda")
contrasts<-c("45","90")
par(mfrow=c(1,2))
for(c in contrasts){
  # Our VULCAN TF enrichment
  if(c=="45"){
    vulcannes<-vobj_tcga_45$msviper$es$nes
  }
  if(c=="90"){
    vulcannes<-vobj_tcga_90$msviper$es$nes
  }
  # ChIPEnrich pathways + TFs
  rawchipenrich<-read.delim(paste0("results/chipenrichr/",c,"_p1_UP_results.tab"),as.is=TRUE)
  chipenrich_tfs<-rawchipenrich[rawchipenrich[,1]=="Transcription Factors",]
  chipenrich_pathways<-rawchipenrich[rawchipenrich[,1]!="Transcription Factors",]
  t1<-gsub(" ","_",paste0(toupper(chipenrich_pathways[,1]),"_",toupper(chipenrich_pathways[,3])))
  t2<-chipenrich_pathways[,"FDR"]
  chipenrich_pathways<-setNames(t2,t1)

  # TF comparison
  cetfs<-setNames(chipenrich_tfs[,"FDR"],gsub("_.+","",chipenrich_tfs[,"Description"]))
  vutfs<-vulcannes[!is.na(vulcannes)]
  vutfs<-vutfs[vutfs>0]
  common<-intersect(names(vutfs),names(cetfs))
```

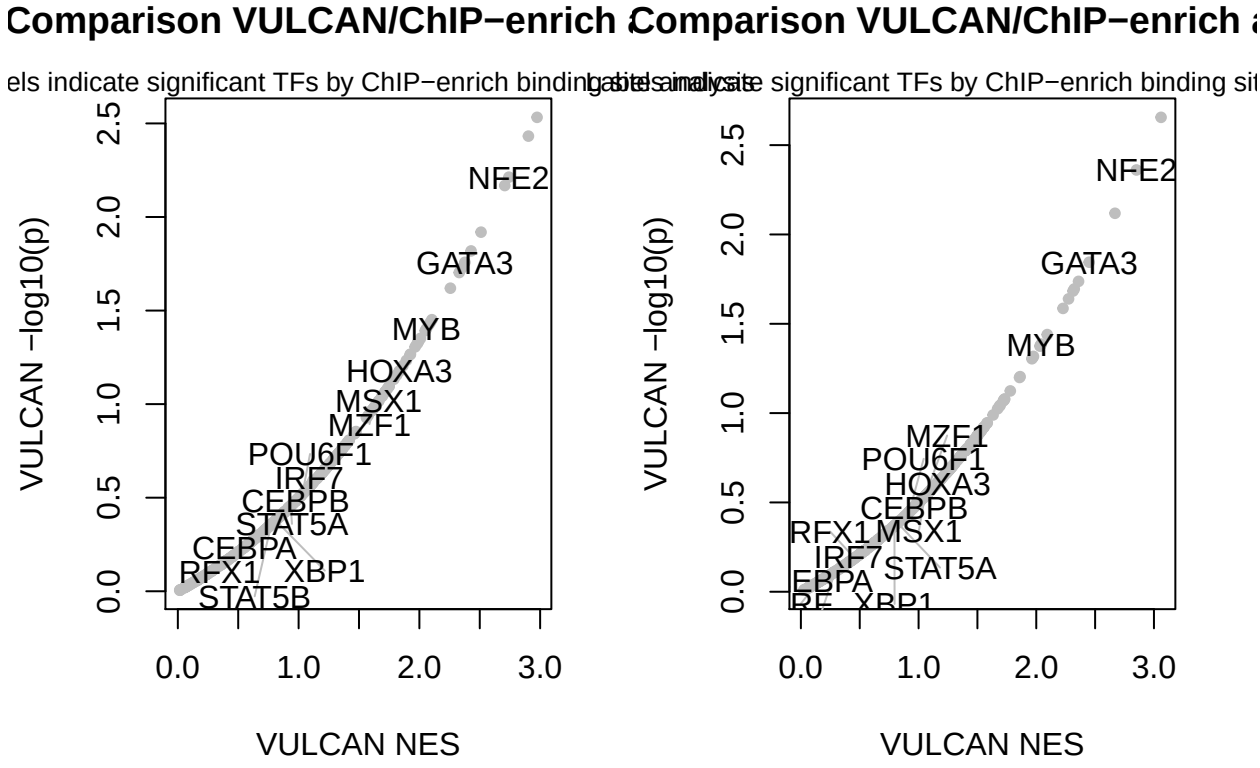


Figure 38: Figure S34. Comparison of results from the VULCAN and ChIP-enrich methods

```
  plot(vutfs,-log10(z2p(vutfs)),xlab="VULCAN NES",ylab="VULCAN -log10(p)",pch=20,col="grey",
       main=paste0("Comparison VULCAN/ChIP-enrich at ",c)
  )
  mtext("Labels indicate significant TFs by ChIP-enrich binding site analysis",cex=0.8)
  set.seed(1)
  textplot2(vutfs[common],-log10(z2p(vutfs))[common],common,new=FALSE)
}
```

```
par(mfrow=c(1,1))
```

```
### Comparison: VULCAN vs. ISMARA
ismara<-t(read.delim("results/ismara/ismara_report/activity_table",as.is=TRUE))
vulcan45<-vobj_tcga_45$msviper$es$nes
vulcan90<-vobj_tcga_90$msviper$es$nes

# Compare VULCAN (x axis) at 45 and 90 minutes with ISMARA (yaxis) average + sd

# Prepare ismara values
i45<-ismara[,grep("45",colnames(ismara))]
i90<-ismara[,grep("90",colnames(ismara))]
i45_mean<-apply(i45,1,mean)
i45_sd<-apply(i45,1,sd)
i90_mean<-apply(i90,1,mean)
i90_sd<-apply(i90,1,sd)

common<-intersect(names(vulcan45),rownames(ismara))
```

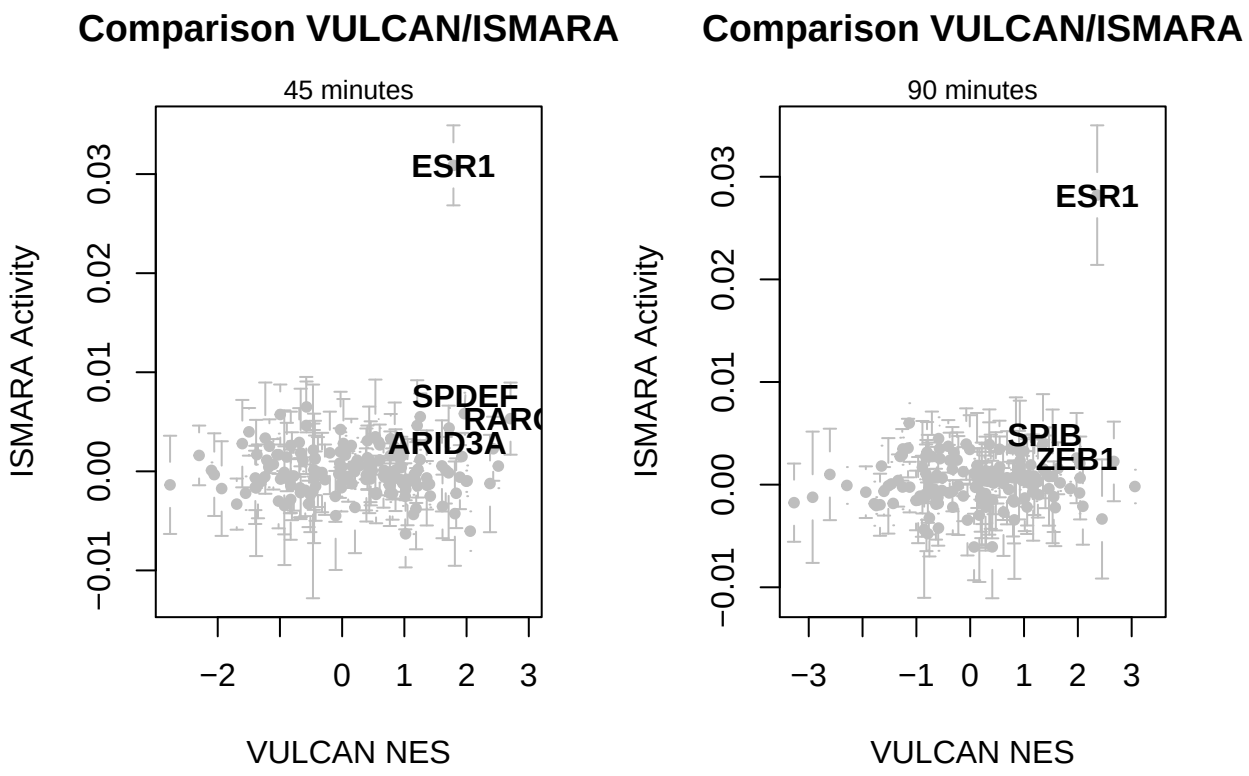


Figure 39: Figure S35. Comparison of results from the VULCAN and ISMARA methods

```
length(common) # 148
```

```
## [1] 148
```

```
# Plots
par(mfrow=c(1,2))

# 45 minutes
x<-vulcan45[common]
y<-i45_mean[common]
uiw<-i45_sd[common]
top<-intersect(names(sort(x,dec=TRUE))[1:20],names(sort(y,dec=TRUE))[1:20])
plotCI(x,y,uiw=uiw,xlab="VULCAN NES",ylab="ISMARA Activity",main="Comparison VULCAN/ISMARA",pch=20,col=
mtext("45 minutes",cex=0.8)
textplot2(x[top],y[top],words=top,new=FALSE,font=2)
# 90 minutes
x<-vulcan90[common]
y<-i90_mean[common]
uiw<-i90_sd[common]
top<-intersect(names(sort(x,dec=TRUE))[1:25],names(sort(y,dec=TRUE))[1:25])
plotCI(x,y,uiw=uiw,xlab="VULCAN NES",ylab="ISMARA Activity",main="Comparison VULCAN/ISMARA",pch=20,col=
mtext("90 minutes",cex=0.8)
textplot2(x[top],y[top],words=top,new=FALSE,font=2)
```

Testing VULCAN on different datasets

We will show how VULCAN can be used on more independent ChIP-Seq datasets, arising from patient data and different sample types.

Breast Cancer Xenografts

We tested VULCAN on another ChIP-Seq dataset (available on GEO series GSE110824). We used a TCGA-derived breast cancer network.

```
list_eg2symbol<-as.list(org.Hs.egSYMBOL[mappedkeys(org.Hs.egSYMBOL)])
fname<-"results/009_vobj_xeno.rda"


if(!file.exists(fname)){
  ctcf<-readRDS("chipseq/holdingNAR/ctcf.rds")
  pdx<-readRDS("chipseq/holdingNAR/pdx.rds")

  # Build input object
  peakcounts<-matrix(NA,nrow=nrow(pdx$peaks[[1]]),ncol=nrow(pdx$samples)+3)
  colnames(peakcounts)<-c("Chr","Start","End",pdx$samples$SampleID)
  peakcounts[,"Chr"]<-pdx$peaks[[1]]$Chr
  peakcounts[,"Start"]<-pdx$peaks[[1]]$Start
  peakcounts[,"End"]<-pdx$peaks[[1]]$End
  peakcounts<-as.data.frame(peakcounts)
  peakrpkms<-peakcounts
  for(i in 1:nrow(pdx$samples)){
    peakcounts[,i+3]<-pdx$peaks[[i]]$Reads
    peakrpkms[,i+3]<-pdx$peaks[[i]]$RPKM
  }
  samples<-list(A=c("PDX01","PDX04"),B=c("PDX03","PDX05"),X="PDX02")
  vobj<-list(peakcounts=peakcounts,samples=samples,peakrpkms=peakrpkms)

  # And then standard VULCAN pipeline
  vobj<-vulcan.annotate(vobj,lborder=-10000,rborder=10000,method='sum')
  vobj<-vulcan.normalize(vobj)
  names(vobj$samples)
  vobj_xeno<-viper(vobj$normalized,regulon=tcga_regulon)
  rownames(vobj_xeno)<-list_eg2symbol[rownames(vobj_xeno)]
  save(vobj_xeno,file=fname)
  rm(vobj,ctcf,pdx)
} else {
  load(fname)
}
```

We show again a co-activity pattern of ESR1 with FOXA1 and GATA3, and an inverse correlation between ESR1 and FOXC1.

```
sometfs<-c("ESR1","FOXA1","GATA3","FOXC1")
vobj_xeno<-vobj_xeno[,c("PDX01","PDX02","PDX03","PDX04","PDX05")]
xenoidconverter<-setNames(c("AB555B","VHI0244o2","AB580","STG195","V0980U"),c("PDX01","PDX02","PDX03","
plot(vobj_xeno[sometfs[1],],type="l",lwd=4,ylab=c("Relative NES in sample"),xaxt="n",main="Xenograft da
axis(1,at=c(1:ncol(vobj_xeno)),labels=xenoidconverter[colnames(vobj_xeno)],font=2)
lines(vobj_xeno[sometfs[2],],type="l",lwd=4,col=2,lty=2)
```

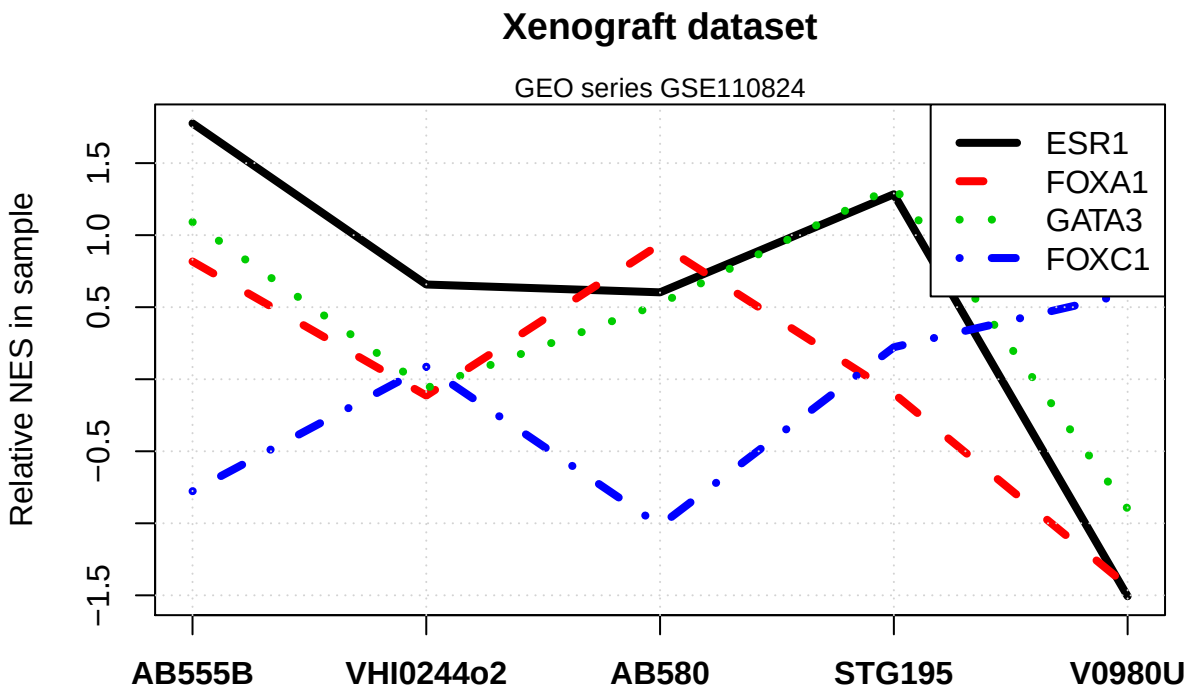


Figure 40: Figure S36. VULCAN Activity scores for a few TFs extracted from the ER-targeted ChIP-Seq Xenograft dataset GSE110824

```
lines(vobj_xeno[sometfs[3],],type="l",lwd=4,col=3,lty=3)
lines(vobj_xeno[sometfs[4],],type="l",lwd=4,col=4,lty=4)
grid()
legend("topright",legend=sometfs,bg="white",col=1:4,lty=1:4,lwd=4)
mtext("GEO series GSE110824",cex=0.8)
```

Prostate Cancer Data

We tested VULCAN on patient ChIP-Seq samples from human prostate cancer cell lines. We used a TCGA-derived prostate cancer network, available on the *aracne.networks* Bioconductor package. We obtained the raw data from the Sahu et al., 2013 Cancer Research study available on GEO series GSE39880. This dataset focuses on Androgen Receptor (AR) binding events in LNCaP-1F5 prostate-derived cells treated with CPA, RU486 and DHT. Reads were aligned using Bowtie2 and peaks were called using Macs 1.4. Only samples with biological replicates in the dataset are shown.

```
# Import
sheetfile<-"chipseq/prostateAR/sheetmini.csv"
fname<-"results/010_input_prostateAR.rda"
if(!file.exists(fname)){
  vobj<-vulcan.import(sheetfile)
  save(vobj,file=fname)
} else {
  load(fname)
}

fnamm<-"results/010_vulcan_prostateAR.rda"
if(!file.exists(fnamm)){
```

```
  samples<-list(AR_CPA=c("AR_CPA_1","AR_CPA_2"),AR_RU486=c("AR_RU486_1","AR_RU486_2"),AR_DHT=c("AR_DHT_
  vobj<-list(peakcounts=vobj$peakcounts,samples=samples,peakrpkms=vobj$peakrpkms)


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

  # Normalization
  vobj_prostateAR<-vulcan.normalize(vobj)
  save(vobj_prostateAR,file="results/010_vobj_prostateAR.rda")

  # Obtain prostate carcinoma (prad) regulon from bioconductor
  library(aracne.networks)
  data(regulonprad)

  # Vulcan inference
  names(vobj_prostateAR$samples)
  vobj_prostateAR<-viper(vobj_prostateAR$normalized,regulon=regulonprad)
  rownames(vobj_prostateAR)<-list_eg2symbol[rownames(vobj_prostateAR)]
  save(vobj_prostateAR,file=fnamm)
} else {
  load(fnamm)
}
```

The bar plots show the relative VULCAN Normalized Enrichment Score calculated on absolute peak intensities after treating cells with Dihydrotestosterone (DHT) and partial AR modulators cyproterone acetate (CPA) and Mifepristone (RU486). FOXA1 network binding is higher in presence of the strong AR recruiter DHT, partially confirming the results of the original study. Two replicates for each treatment were produced and are reported in matching colors.

```
par(las=2)
toplot<-vobj_prostateAR["FOXA1",]
names(toplot)<-gsub("AR_","",names(toplot))
names(toplot)<-gsub("_"," ",names(toplot))
barplot(toplot,cex.names=0.8,col=c(2,2,3,3,4,4),ylab="FOXA1 Vulcan Score")
par(las=1)
title("AR ChIP-Seq in Prostate Cells")
mtext("GEO series GSE39880",cex=0.8)
```

Characterize GHRL2

We will perform a Fisher's Exact one tail test analysis using the MSIGDB database, to collect clues on the function of this transcriptional regulator. Note: the hypergeometric R implementation for the one-tailed Fisher test, as described here: https://mengnote.blogspot.it/2012/12/calculate-correct-hypergeometric-p.html (Results are identical to a Fisher test, but faster to compute).

```
fnamm<-"results/011_grhl2_fet.rda"
if(!file.exists(fnamm)){
  load("data/msigdb.v6.1.entrez.rda")
  targets_grhl2<-names(tcga_regulon[["79977"]]$tfmode)

  ### Calculate Contigency Tables
  contingencyFunction<-function(set,list){
```

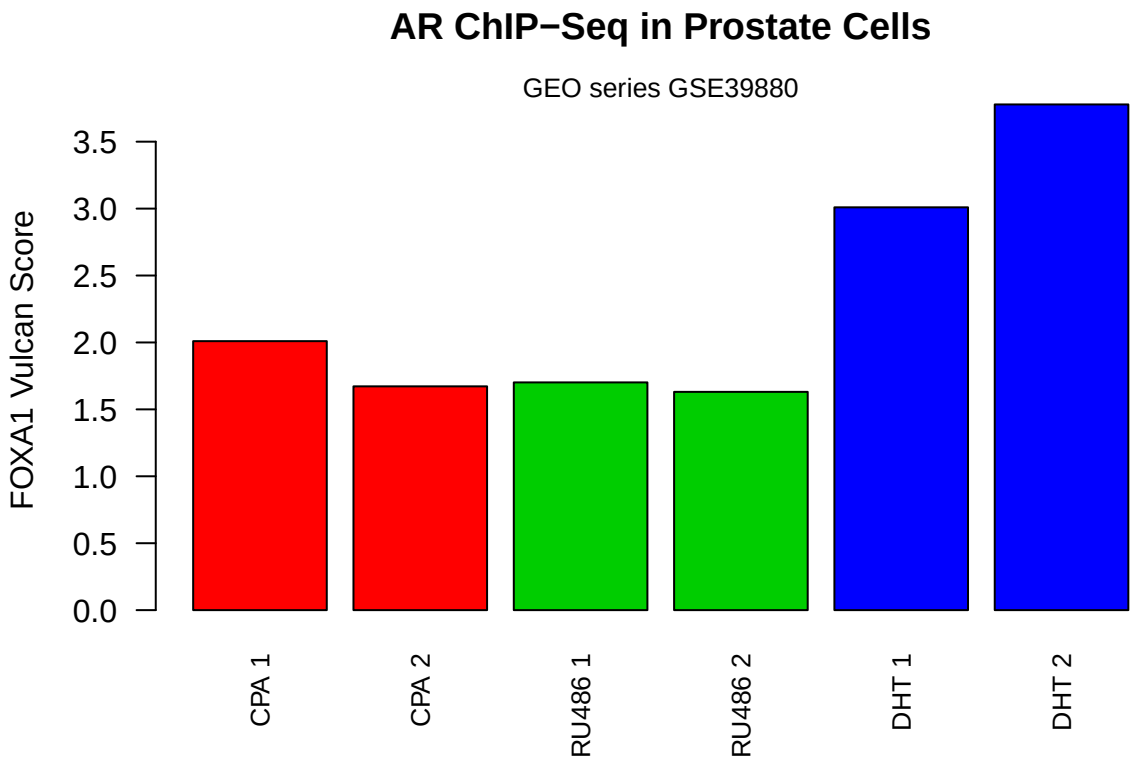


Figure 41: Figure S37. VULCAN Activity scores for FOXA1 in Prostate cell lines (dataset GSE39880)

```
  if(!exists("universegenelist")){
    universegenelist<<-unique(unlist(list))
  }
  out<-t(sapply(list,function(x){
    ul<-intersect(x,set)
    ur<-setdiff(set,ul)
    dl<-setdiff(x,ul)
    dr<-setdiff(universegenelist,c(ul,ur,dl))
    return(c(length(ul),length(ur),length(dl),length(dr)))
  }))
  return(out)
}
grhl2_ctables<-contingencyFunction(targets_grhl2,msigdb)

grhl2_fet<-apply(grhl2_ctables,1,function(f){
  w<-f[1]
  x<-f[2]
  y<-f[3]
  z<-f[4]
  p<-phyper(w-1,x+w,z+y,y+w,lower.tail=FALSE)
})



save(grhl2_fet,file=fnamm)
```

```
  rm(msigdb)
} else {
  load(fnamm)
}
```

```
# Write the output NICELY
classnames<-setNames(
  c(
    "C1 positional gene sets",
    "C2 chemical and genetic perturbations",
    "C2 BioCarta gene sets",
    "C2 KEGG gene sets",
    "C2 Reactome gene sets",
    "C2 Canonical pathways",
    "C3 microRNA targets",
    "C3 transcription factor targets",
    "C4 cancer gene neighborhoods",
    "C4 cancer modules",
    "C5 GO biological process",
    "C5 GO cellular component",
    "C5 GO molecular function",
    "C6 oncogenic signatures",
    "C7 immunologic signatures",
    "H hallmark gene sets"
  ),
  c("c1_all","c2_cgp","c2_cpbiocarta","c2_cpkegg","c2_cpreactome","c2_cp","c3_mir","c3_tft",
    "c4_cgn","c4_cm","c5_bp","c5_cc","c5_mf","c6_all","c7_all","h_all")
)


# Print Enrichment
fet<-grhl2_fet
classes<-unique(gsub(";_;.+","",names(fet)))
append<-FALSE
class<-classes[2]
subfet<-fet[grep(paste0(class,";_;"),names(fet))]
names(subfet)<-gsub(".+;_;","",names(subfet))
subfet<-sort(subfet,dec=FALSE)
subfet<-cbind(names(subfet),subfet,p.adjust(subfet,method="bonferroni"))
colnames(subfet)<-c("Gene Set","P-value","Q-value")
subfet[,2]<-signif(as.numeric(subfet[,2]),3)
subfet[,3]<-signif(as.numeric(subfet[,3]),3)
rownames(subfet)<-NULL


grid.newpage()
grid.table(subfet[1:10,])
```

Overlap with Motif analysis

We compare our analysis with a direct binding strategy. In order to do so, we directly compare the gene networks used by VULCAN with transcription factor targets gene sets that share upstream cis-regulatory

Gene Set	P−value	Q−valu
HARAFE_BREAST_CANCER_LUMINAL_VS_MESENCHYMAL_UP	4.46e−14	1.52e−
NIKOLSKY_BREAST_CANCER_8Q23_Q24_AMPLICON	3.21e−08	0.0001
LIM_MAMMARY_STEM_CELL_UP	3.8e−08	0.000
WAKABAYASHI_ADIPOGENESIS_PPARG_RXRA_BOUND_8D	8.59e−08	0.0002
SWEET_LUNG_CANCER_KRAS_DN	1.08e−07	0.0003
PILON_KLF1_TARGETS_DN	1.15e−07	0.0003
ONKEN_UVEAL_MELANOMA_UP	1.32e−07	0.0004
BOQUEST_STEM_CELL_DN	3.73e−07	0.001
COLDREN_GEFITINIB_RESISTANCE_DN	6e−07	0.002
ACEVEDO_LIVER_CANCER_DN	9.14e−07	0.003

Figure 42: Table S7. 10 gene sets from the MsigDB chemical/genetic perturbetion set, which are most overlapping with GRHL2 targets identified by our network analysis

motifs which can function as potential transcription factor binding sites. This motif analysis is based on work by Xie et al. 2005 and processed by the MsigDB project (C3 collection).

```
load("data/msigdb.v6.1.entrez.rda")

# VULCAN score correlation example: GATA3 vs ESR1
load("results/001_vobj.rda") # vobj
fname<-"results/012_vulcanscores.rda"
if(!file.exists(fname)){
  vulcanscores<-viper(vobj$normalized,regulon=tcga_regulon)
  save(vulcanscores,file=fname)
} else {
  load(fname)
}
X<-vulcanscores[list_symbol2eg["ESR1"],]
Y<-vulcanscores[list_symbol2eg["GATA3"],]
plot(X,Y,xlab="ESR1 VULCAN score",ylab="GATA3 VULCAN score",main="VULCAN score in our dataset",pch=NA)
text(X,Y,labels=colnames(vulcanscores))
grid()
abline(lm(Y~X)$coef)
scc<-signif(cor(X,Y,method="s"),3)
mtext(paste0("Spearman correlation: ",scc),cex=0.8)
```

```
# Motif-based targets overlap example: GATA3 vs. ESR1
motifdb<-msigdb[grep("c3_tft;_;",names(msigdb))]
grep("GATA3",names(motifdb),value=TRUE)
```

```
## [1] "c3_tft;_;GATA3_01"
```

```
grep("ER",names(motifdb),value=TRUE)
```

```
## [1] "c3_tft;_;ER_Q6"           "c3_tft;_;ERR1_Q2"
## [3] "c3_tft;_;NERF_Q2"         "c3_tft;_;P53_DECAMER_Q2"
```

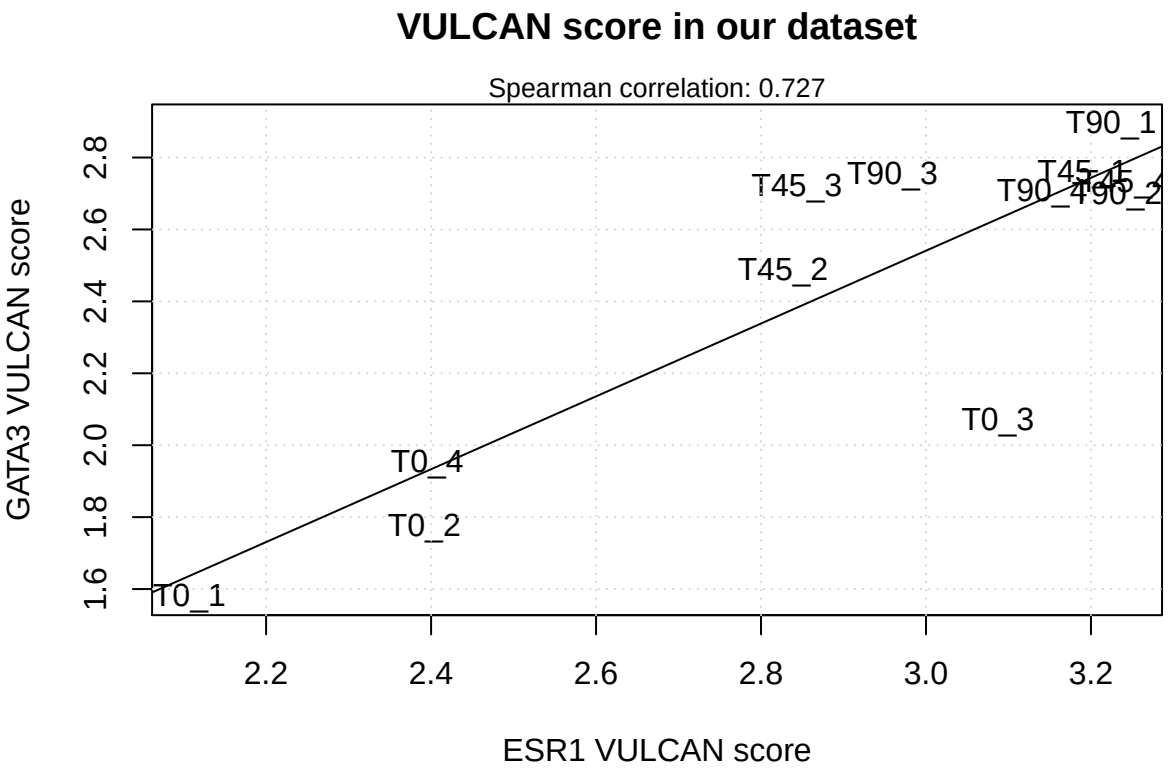


Figure 43: Figure S38. Example: VULCAN scores of GATA3 and ESR1 in our dataset. Individual samples are indicated.

```
## [5] "c3_tft;_;ER_Q6_01"        "c3_tft;_;ER_Q6_02"
## [7] "c3_tft;_;TGACCTY_ERR1_Q2"
```

```
ER_Targets<-msigdb[["c3_tft;_;ER_Q6"]]
GATA3_Targets<-msigdb[["c3_tft;_;GATA3_01"]]
# Significance of overlap (Fisher Test)
ul<-intersect(GATA3_Targets,ER_Targets)
ur<-setdiff(GATA3_Targets,ul)
dl<-setdiff(ER_Targets,ul)
setp<-fet<-fisher.test(rbind(c(length(ul),length(ur)),c(length(dl),20000-length(ul)-length(ur)-length(dl
# Plot it
library(VennDiagram)
v<-venn.diagram(list(GATA3_Targets=GATA3_Targets, ER_Targets=ER_Targets),
                fill = c("orange", "blue"),
                alpha = c(0.5, 0.5), cat.cex=1,
                filename=NULL,lty=2,
                #main=paste0("FET p=",signif(setp,4)),
                cex=0.6,rotation.degree = 0,main.pos=c(0.5,0)
)
grid.newpage()
grid.draw(v)
```

We then decided to compare TF overlap according to VULCAN score and a classic approach: overlap of putative target genes with promoters carrying canonical and TF specific binding motifs. There is no correlation between the two approaches.

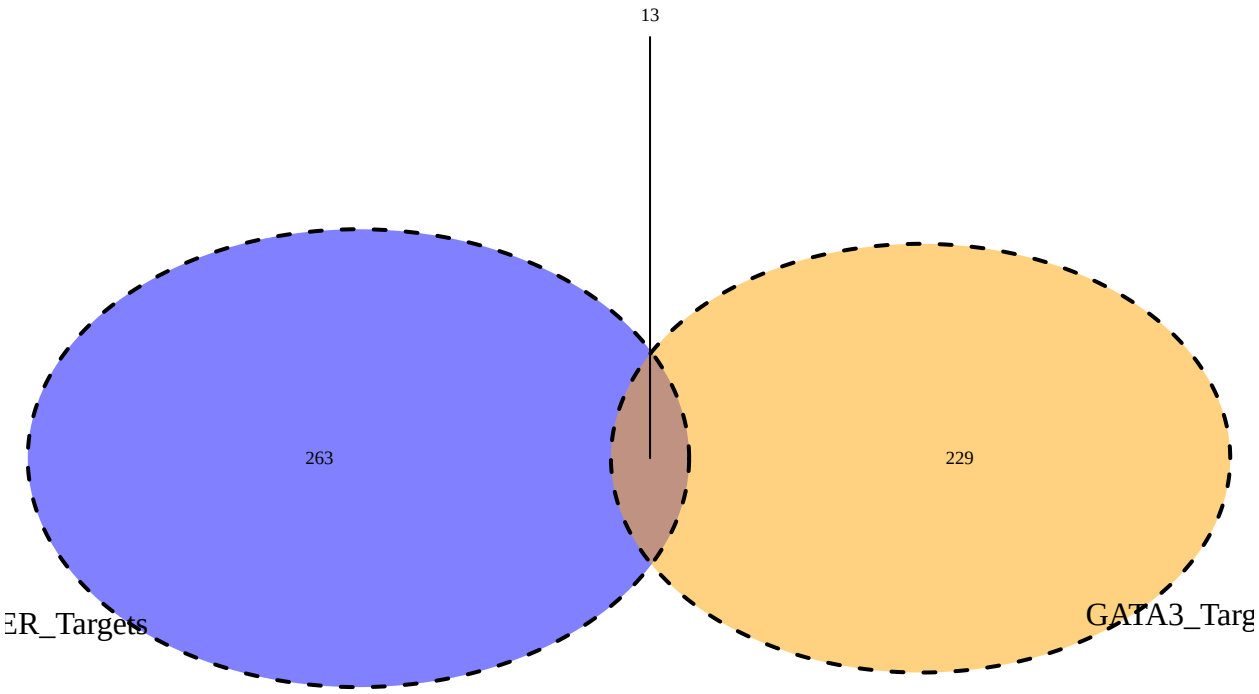


Figure 44: Figure S39. Example of target intersection between GATA3 and ER according to the MSigDB database of canonical TF-specific motifs in putative target gene promoters

```
# Enrichment of target overlaps with targets defined by motif analysis

### any2entrez function
any2entrez<-function(x){
  tab<-AnnotationDbi::select(org.Hs.eg.db, keys=x, columns=c("ENTREZID"), keytype="ALIAS")
  symbols<-tab[,1]
  entrez<-tab[,2]
  dups<-which(duplicated(symbols))
  symbols<-symbols[-dups]
  entrez<-entrez[-dups]
  out<-setNames(entrez,symbols)
  return(out)
}

# Get TFs from the msigdb motif database
submotifdb<-motifdb[grep("UNKNOWN",names(motifdb),invert=TRUE)]
names(submotifdb)<-gsub("c3_tft;_;","",names(submotifdb))
names(submotifdb)<-gsub("ER_","ESR1_",names(submotifdb))
names(submotifdb)[1:441]<-sapply(strsplit(names(submotifdb)[1:441],"_"),function(x){x[1]})
names(submotifdb)[441:500]<-sapply(strsplit(names(submotifdb)[441:500],"_"),function(x){x[2]})
names(submotifdb)<-any2entrez(names(submotifdb))
submotifdb<-submotifdb[!is.na(names(submotifdb))]
tfs_with_motifs<-names(submotifdb)
tfs_with_vulcan<-rownames(vulcanscores)
tfs_with_both<-intersect(tfs_with_motifs,tfs_with_vulcan)

# Compare VULCAN correlation with MOTIF overlap
```

```
corvulcan<-cor(t(vulcanscores[tfs_with_both,]),method="s")
fetmotif<-matrix(NA,nrow=nrow(corvulcan),ncol=ncol(corvulcan),dimnames=dimnames(corvulcan))
for(i in 1:nrow(corvulcan)){
  for(j in (i+1):nrow(corvulcan)){
    if(j>nrow(corvulcan)|j==i){
      next
    }
    targets1<-submotifdb[[rownames(corvulcan)[i]]]
    targets2<-submotifdb[[rownames(corvulcan)[j]]]
    JI<-length(intersect(targets1,targets2))/length(union(targets1,targets2))
    fetmotif[i,j]<-JI

  }
}
```

```
# Plot the Motif overlap JI with the vulcan-score correlation between TF pairs
x<-corvulcan[upper.tri(corvulcan,diag=FALSE)]
y<-fetmotif[upper.tri(fetmotif,diag=FALSE)]

plot(x,y,xlab="SCC VULCAN score",ylab="Jaccard Index of Motif Overlap",main="TF pairs",pch=20)
minpair<-which(fetmotif==max(fetmotif,na.rm=TRUE),arr.ind=TRUE)
tf1<-list_eg2symbol[[rownames(corvulcan)[minpair[1,1]]]]
tf2<-list_eg2symbol[[rownames(corvulcan)[minpair[1,2]]]]
text(corvulcan[minpair[1,1],minpair[1,2]],fetmotif[minpair[1,1],minpair[1,2]],labels=paste0(tf1,"-",tf2

tf1<-list_symbol2eg[["ESR1"]]
tf2<-list_symbol2eg[["GATA3"]]
text(corvulcan[tf2,tf1],fetmotif[tf2,tf1],labels=paste0(list_eg2symbol[[tf1]],"-",list_eg2symbol[[tf2]]]
```

```
rm(msigdb)
```

Technical Session Info

The following code describes the R environment used to generate this document and will help making it fully reproducible should there be future updates in any of the packages.

```
sessionInfo()
```

```
## R version 3.5.0 (2018-04-23)
## Platform: x86_64-apple-darwin15.6.0 (64-bit)
## Running under: OS X El Capitan 10.11.6
##
## Matrix products: default
## BLAS: /Library/Frameworks/R.framework/Versions/3.5/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/3.5/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_GB.UTF-8/en_GB.UTF-8/en_GB.UTF-8/C/en_GB.UTF-8/en_GB.UTF-8
##
## attached base packages:
##  [1] stats4    parallel  grid      stats     graphics  grDevices utils
##  [8] datasets  methods   base
##
```

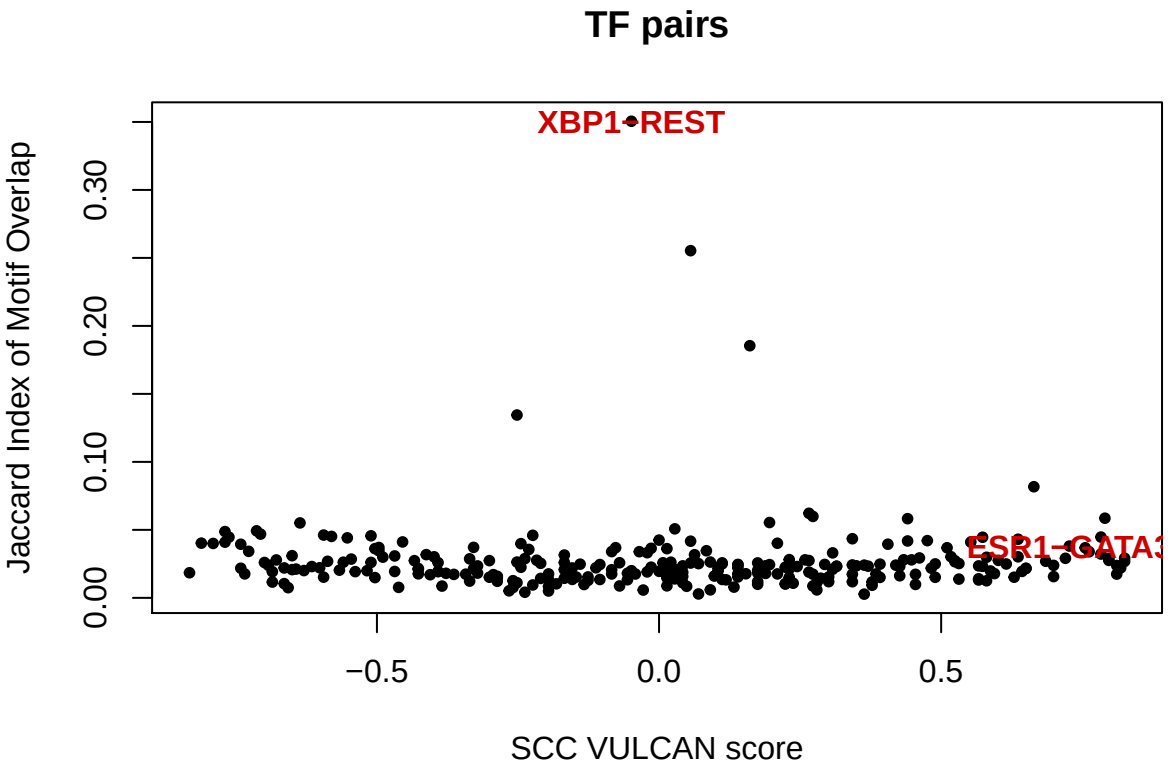


Figure 45: Figure S40. TF pairs compared in terms of VULCAN score Spearman Correlation Coefficient in our ER dataset and in terms of Jaccard Index of motif-based target intersection according to the MSigDB C3 collection

```
## other attached packages:
##  [1] GeneNet_1.2.13
##  [2] fdrtool_1.2.15
##  [3] longitudinal_1.1.12
##  [4] corpcor_1.6.9
##  [5] gridExtra_2.3
##  [6] org.Hs.eg.db_3.6.0
##  [7] gplots_3.0.1
##  [8] vulcan_1.4.0
##  [9] locfit_1.5-9.1
## [10] DiffBind_2.8.0
## [11] SummarizedExperiment_1.10.1
## [12] DelayedArray_0.6.1
## [13] BiocParallel_1.14.1
## [14] matrixStats_0.53.1
## [15] viper_1.16.0
## [16] zoo_1.8-2
## [17] TxDb.Hsapiens.UCSC.hg19.knownGene_3.2.2
## [18] GenomicFeatures_1.32.0
## [19] AnnotationDbi_1.42.1
## [20] Biobase_2.40.0
## [21] ChIPpeakAnno_3.14.0
## [22] VennDiagram_1.6.20
## [23] futile.logger_1.4.3
## [24] GenomicRanges_1.32.3
## [25] GenomeInfoDb_1.16.0
## [26] Biostrings_2.48.0
## [27] XVector_0.20.0
## [28] IRanges_2.14.10
## [29] S4Vectors_0.18.3
## [30] BiocGenerics_0.26.0
##
## loaded via a namespace (and not attached):
##   [1] backports_1.1.2          GOstats_2.46.0
##   [3] Hmisc_4.1-1              plyr_1.8.4
##   [5] lazyeval_0.2.1           GSEABase_1.42.0
##   [7] splines_3.5.0            BatchJobs_1.7
##   [9] ggplot2_3.0.0.9000       amap_0.8-16
##  [11] digest_0.6.16            BiocInstaller_1.30.0
##  [13] ensembldb_2.4.1          htmltools_0.3.6
##  [15] GO.db_3.6.0              gdata_2.18.0
##  [17] csaw_1.16.0              magrittr_1.5
##  [19] checkmate_1.8.5          memoise_1.1.0
##  [21] BBmisc_1.11              BSgenome_1.48.0
##  [23] cluster_2.0.7-1          mixtools_1.1.0
##  [25] limma_3.36.1             annotate_1.58.0
##  [27] wordcloud_2.6            systemPipeR_1.14.0
##  [29] prettyunits_1.0.2        colorspace_1.3-2
##  [31] blob_1.1.1               ggrepel_0.8.0
##  [33] dplyr_0.7.5              crayon_1.3.4
##  [35] RCurl_1.95-4.10          graph_1.58.0
##  [37] genefilter_1.62.0        bindr_0.1.1
##  [39] brew_1.0-6               survival_2.42-3
##  [41] sendmailR_1.2-1          glue_1.2.0
```

```
##  [43] gtable_0.2.0             zlibbioc_1.26.0
##  [45] seqinr_3.4-5             Rgraphviz_2.24.0
##  [47] scales_1.0.0             DESeq_1.34.0
##  [49] futile.options_1.0.1     pheatmap_1.0.10
##  [51] DBI_1.0.0                edgeR_3.22.2
##  [53] Rcpp_0.12.18             htmlTable_1.12
##  [55] xtable_1.8-2             progress_1.2.0
##  [57] foreign_0.8-70           bit_1.1-14
##  [59] Formula_1.2-3            AnnotationForge_1.22.0
##  [61] htmlwidgets_1.2          httr_1.3.1
##  [63] RColorBrewer_1.1-2       acepack_1.4.1
##  [65] pkgconfig_2.0.1          XML_3.98-1.11
##  [67] nnet_7.3-12              tidyselect_0.2.4
##  [69] rlang_0.2.2              munsell_0.5.0
##  [71] tools_3.5.0              RSQLite_2.1.1
##  [73] ade4_1.7-11              evaluate_0.10.1
##  [75] stringr_1.3.1            yaml_2.1.19
##  [77] knitr_1.20               bit64_0.9-7
##  [79] caTools_1.17.1           purrr_0.2.5
##  [81] AnnotationFilter_1.4.0   bindrcpp_0.2.2
##  [83] RBGL_1.56.0              formatR_1.5
##  [85] biomaRt_2.36.1           rstudioapi_0.7
##  [87] compiler_3.5.0           curl_3.2
##  [89] e1071_1.7-0              geneplotter_1.58.0
##  [91] tibble_1.4.2             stringi_1.2.3
##  [93] idr_1.2                  lattice_0.20-35
##  [95] ProtGenerics_1.12.0      Matrix_1.2-14
##  [97] multtest_2.36.0          pillar_1.2.3
##  [99] data.table_1.11.4        bitops_1.0-6
## [101] rtracklayer_1.40.3       R6_2.2.2
## [103] latticeExtra_0.6-28      hwriter_1.3.2
## [105] ShortRead_1.38.0         KernSmooth_2.23-15
## [107] lambda.r_1.2.3           MASS_7.3-50
## [109] gtools_3.5.0             assertthat_0.2.0
## [111] DESeq2_1.20.0            Category_2.46.0
## [113] rprojroot_1.3-2          rjson_0.2.20
## [115] regioneR_1.12.0          GenomicAlignments_1.16.0
## [117] Rsamtools_1.32.0         GenomeInfoDbData_1.1.0
## [119] hms_0.4.2                rpart_4.1-13
## [121] class_7.3-14             rmarkdown_1.10
## [123] segmented_0.5-3.0        base64enc_0.1-3
```