Gene Regulatory Network Finder (GRNF)

Ryne Ramaker

2017-05-11

Overview

The primary function of this package is to provide tools for assessing which genes a DNA associated protien (DAP) likely regulates based on its genome occupancy.

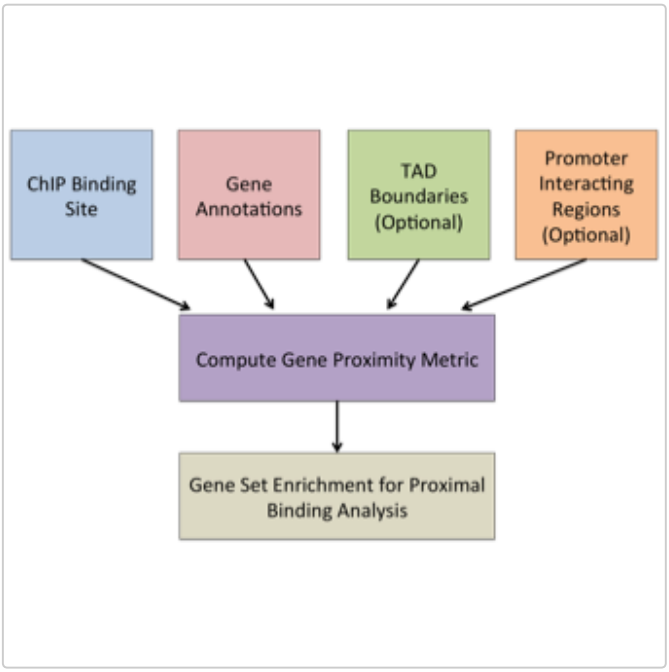

Figure 1. Overview of package workflow.

The primary function *GeneToPeakDist* computes the distance from gene transcription start sites (TSS) to the nearest DAP binding site. This function can also calculate weighted distances based on chromatin structure information such as topologically associated domains or promoter interactions. A weighted gene proximity metric is computed for each gene as follows:

If a DAP binding site location (peak end closest to a gene's TSS) can be represented as $d_i, \forall i \in [1, 2, 3...n]$ where n is the number of DAP binding sites on a gene's chromosome, a binding proximity metric for each gene, q, can be computed as:

$$G_q = argmin(\frac{|T - d_i| \times E}{1 + K})$$

where

$$K = L \times F$$

Where G_q is the weighted gene proximity metric and T is the gene's transcription start site. E is a TAD penalty which is equal to one if d falls within a gene's TAD and a constant greater than one if d falls outside a gene's TAD. K is a promoter interaction bonus which consists of the product of L, a non-zero constant if d falls within a genomic region that has been shown to interact with gene's promoter or zero if d does not fall in a promoter interacting region, and F, the relative interaction frequency for the promoter interacting region. This is illustrated for an imaginary gene below:

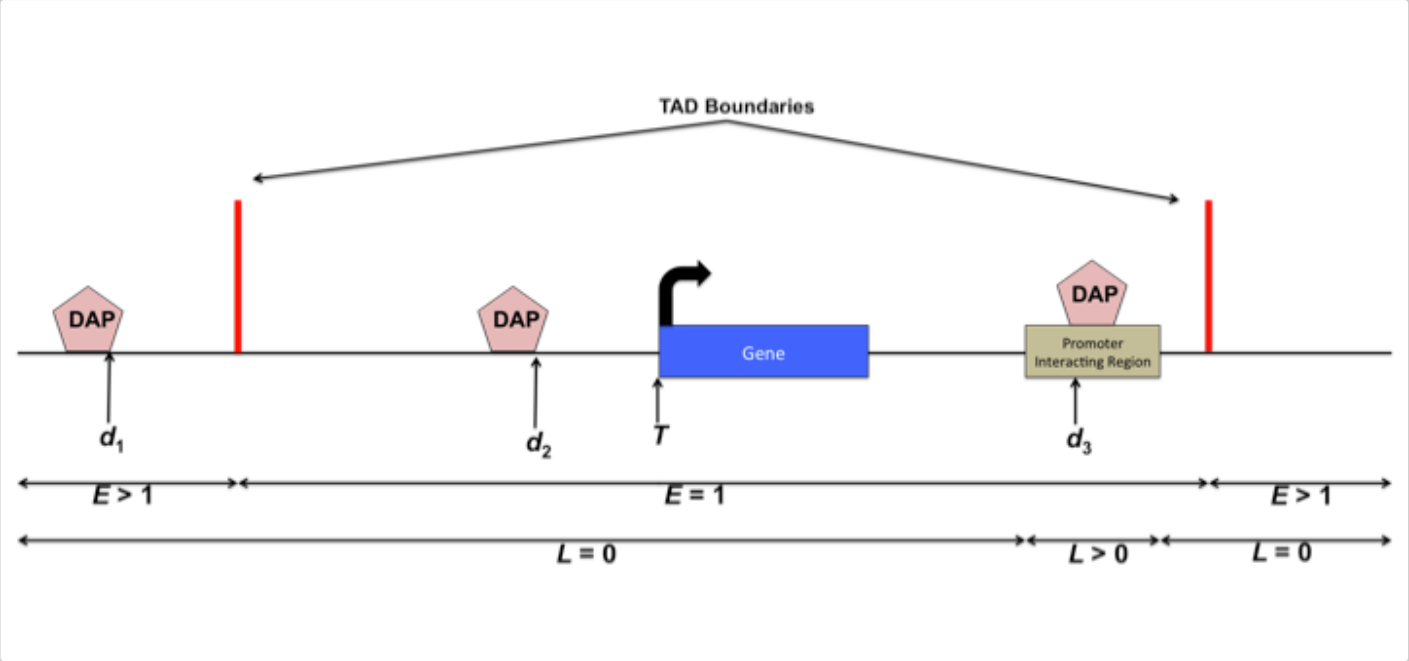

Figure 2. Visualization of GeneToPeakDist function variables

The second function *FindPathwayEnrichment* will assess enrichment for proximal binding to Reactome pathway gene sets or a user inputed gene set as illustrated below:

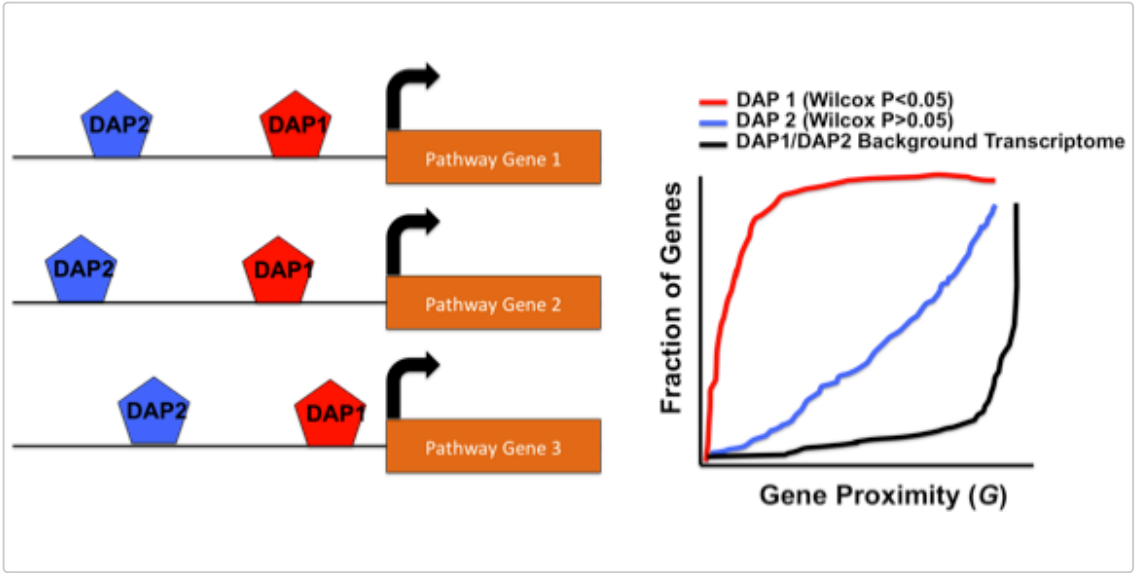

Figure 3. Pathway analysis workflow.

Installation

First, ensure you have the pacakge 'GenomicRanges' installed:

```
source("https://bioconductor.org/biocLite.R")
biocLite("GenomicRanges")
```

Next, there are multiple ways to install this package. The first involves downloading the binary source file from my github page (https://github.com/rramaker/GRNF) and installing the package manually:

```
install.packages("/PathToDownloadedFile/GRNF_0.1.0.tgz")
```

The package can be directly installed from github with the devtools package

```
library(devtools)
install_github("rramaker/GRNF")
```

Alternatively the package can be downloaded from the CRAN repository. ##Coming Soon!

```
install.packages("GRNF")
```

Once installed the functions from GRNF can be made available in your current working environment with the library() command:

```
library(GRNF)
```

Formatting data

ChIP-seq peak input (BED file)

The primary required input is a dataframe in BED format (https://genome.ucsc.edu/FAQ/FAQformat.html#format1) with each row representing the genomic location of a unique ChIP-seq peak. There are only three columns. The first column should indicate the chromsome of a binding site. The second and third column should indicate the start and stop position of each binding site. Additional columns after these first three are acceptalbe. An example publically available ChIP-seq data set[1] has been provided with the package and can be accessed as shown below:

```
data("GM12878_BATF_ChIP")
head(GM12878_BATF_ChIP[,1:3])
```

```
##      V1        V2        V3
## 1 chr19  13210804  13211092
## 2 chr20  17806838  17807133
## 3  chr6 108885475 108885784
## 4  chr3  45906242  45906558
## 5  chr5 140090648 140090977
## 6  chr9  98614900  98615216
```

Gene position input (GTF file)

The second required input is a dataframe in GTF format (http://www.ensembl.org/info/website/upload/gff.html) with each row representing the genomic location of an annotated gene. There are specifically five required

columns that will queried. The first column should indicate the chromosome position of each gene. The format of this column should match the chromosome column of the ChIP-seq peak input described above (i.e. “chr1” vs. “1”). The third and fourth columns should indicate the start and stop position of each gene. The seventh column should contain strand information and the ninth column should be an “attribute” column that at minimum contains the ENSEMBL ID of each gene. An example hg19/GRCh37 ENSEMBL GTF file has been provided with the package and can be accessed as shown below:

```
data("Homo_sapiens.GRCh37.82.chr.gtf")
head(Homo_sapiens.GRCh37.82.chr.gtf)
```

```
##       V1              V2   V3     V4     V5 V6 V7 V8
## 110 chr1 ensembl_havana gene  69091  70008  .  +  .
## 141 chr1        ensembl gene 134901 139379  .  -  .
## 277 chr1 ensembl_havana gene 367640 368634  .  +  .
## 374 chr1 ensembl_havana gene 621059 622053  .  -  .
## 511 chr1        ensembl gene 738532 739137  .  -  .
## 606 chr1        ensembl gene 818043 819983  .  +  .
##
V9
## 110  gene_id "ENSG00000186092"; gene_version "4"; gene_name "OR4F5"; gene_source "ensembl_havana";
gene_biotype "protein_coding";
## 141    gene_id "ENSG00000237683"; gene_version "5"; gene_name "AL627309.1"; gene_source "ensembl";
gene_biotype "protein_coding";
## 277 gene_id "ENSG00000235249"; gene_version "1"; gene_name "OR4F29"; gene_source "ensembl_havana";
gene_biotype "protein_coding";
## 374 gene_id "ENSG00000185097"; gene_version "2"; gene_name "OR4F16"; gene_source "ensembl_havana";
gene_biotype "protein_coding";
## 511    gene_id "ENSG00000269831"; gene_version "1"; gene_name "AL669831.1"; gene_source "ensembl";
gene_biotype "protein_coding";
## 606    gene_id "ENSG00000269308"; gene_version "1"; gene_name "AL645608.2"; gene_source "ensembl";
gene_biotype "protein_coding";
```

Topological Associated Domain (TAD) boundary input

Optionally, TAD boundary information can be provided and used to penalize TSS to binding site distances. The required format for TAD boundaries is a dataframe with each row representing a unique TAD. The first column should indicate the chromosome of each TAD (formatted identically to the ChIP-seq BED and gene GTF inputs) and the second and third columns should indicate the TAD boundaries. An example publically available TAD file[2] has been provided with the package and can be accessed as shown below:

```
data("GM12878_TAD")
head(GM12878_TAD)
```

```
##    chr   start     end
## 1 chr1 1646875 1842500
## 2 chr1 2112500 2342500
## 3 chr1 2347500 2608125
## 4 chr1 3763750 4396250
## 5 chr1 4429375 5535000
## 6 chr1 5540000 6022500
```

Promoter interaction input

Optionally, information on which genomic regions interact with gene promoters, as measured by assays such as promoter capture Hi-C (PCHC), can be provided to appropriatley reduce TSS to binding site distances if binding sites are found in strong promoter interacting regions. The required input format consists of five columns. The first three columns should indicate the chromsome (formatted consistently with ChIP-seq BED files and gene GTF files described above), start and stop positions of each "captured" promoter interacting region. The fourth column should provide a metric associated with the interaction frequency of each "captured" region with it's respective promoter "bait". The fifth column should contain the ENSEMBL ID of the gene who's promoter acts as "bait" for an interaction. An example input describing publically available promoter interaction data has been provided with the package and can be accessed as shown below:

```
data("GM12878_PCHC")
head(GM12878_PCHC)
```

```
##      V1      V2      V3       V4              V5
## 1  chr1 1583927 1585571 14.22571 ENSG00000008128
## 2  chr1 1583927 1585571 14.22571 ENSG00000215790
## 3  chr1 1583927 1585571 14.22571 ENSG00000226628
## 4  chr1 1583927 1585571 14.22571 ENSG00000227775
## 5  chr1 1583927 1585571 14.22571 ENSG00000244250
## 6 chr19 1228423 1232133 14.12849 ENSG00000064932
```

Finding ChIP-seq peak to gene TSS distances with GeneToPeakDist

The simplest use of this package is computed the genomic distance between TSS and nearest binding site for a given DNA associated protein and a set of genes. Binding sites should be provided under the *ChIP* flag and gene coordinates should be provided under the *GTF* flag. Specific genes can be specified with the *Genes* flag. This can be performed with example data provided with the package as shown below:

```
data("GM12878_BATF_ChIP")
data("Homo_sapiens.GRCh37.82.chr.gtf")
GeneDists<- GeneToPeakDist(ChIP = GM12878_BATF_ChIP, GTF= Homo_sapiens.GRCh37.82.chr.gtf,
Genes=c("ENSG00000186092", "ENSG00000237683", "ENSG00000235249","ENSG00000185097"))
```

The output of this function is a two column dataframe containing the gene ID in the first column and the distance to the nearest binding site in the second column:

```
head(GeneDists)
```

```
##            GeneID Distance (bp)
## 1 ENSG00000186092        756694
## 2 ENSG00000235249        458145
## 3 ENSG00000237683        686406
## 4 ENSG00000185097        203732
```

Weighted distance metrics can also be computed by penalizing regions of the genome for falling outside of gene's TAD. TAD information should be provided with the *TAD* flag and the magnitude of the penalty for falling outside a gene's TAD is indicated with the *TAD_Penalty* flag. We can expand our previous example as shown below:

```
data("GM12878_BATF_ChIP")
data("Homo_sapiens.GRCh37.82.chr.gtf")
data("GM12878_TAD")
GeneDists<- GeneToPeakDist(ChIP = GM12878_BATF_ChIP, GTF= Homo_sapiens.GRCh37.82.chr.gtf, TAD =
GM12878_TAD, TAD_Penalty = 100)
```

An additional distance weighting that can be perfomed is a bonus for genomic regions known to interact frequently with a gene's promoter. This interaction information should be provided with the *PCHC* flag and the magnitude of the bonus falling into a promoter interacting region is indicated with the *PCHC_Bonus* flag. We can expand our previous example as shown below:

```
data("GM12878_BATF_ChIP")
data("Homo_sapiens.GRCh37.82.chr.gtf")
data("GM12878_TAD")
data("GM12878_PCHC")
GeneDists<- GeneToPeakDist(ChIP = GM12878_BATF_ChIP, GTF= Homo_sapiens.GRCh37.82.chr.gtf, TAD =
GM12878_TAD, TAD_Penalty = 100, PCHC = GM12878_PCHC, PCHC_Bonus = 100)
```

Performing pathway analysis for proximal binding with FindPathwayEnrichment

The second function allows user's to perform pathway enrichment analysis using Reactome (http://www.reactome.org) pathways. This implements a simple one-sided Wilcox test comparing the distances from TSS to nearest binding site of genes within a given pathway to all other genes not found in a pathway with the alternate hypothesis that binding sites occur closer to pathway genes than non-pathway genes. The two column dataframe resulting from the *GeneToPeakDist* function is provided to the *distanceFrame* flag and the minimum number of genes required for inclusion of a pathway is provided under the *minPathwayGenes* flag. This analysis can be performed as shown below:

```
PathwayPs<-FindPathwayEnrichment(distanceFrame = GeneDists, minPathwayGenes = 5)
```

This function returns a vector of P-values resulting from the pathway analysis.

```
head(PathwayPs)
```

```
## REACT_264071 REACT_264164 REACT_264075 REACT_264487 REACT_111118
##  0.005624701  0.006564192  0.004006649  0.014625919  0.007763216
##  REACT_18273
##  0.024889043
```

The user can also input a gene set of interest under the *geneSet* flag for enrichment analysis as shown below:

```
FindPathwayEnrichment(distanceFrame = GeneDists, geneSet=
c("ENSG00000156006","ENSG00000196839","ENSG00000170558","ENSG00000133997","ENSG00000254415"))
```

```
## [1] 0.6521047
```

Improving performance with parallelization

Both functions make use of parallelization with R's parallel package. The number of cores used by the function can be changed with the *numCores* flag in both functions. We have found most computers can readily handle 4 cores (the default), but this can be increased on systems with higher capacity. The number of cores available on a computer can be determined as shown below:

```
library(parallel)
detectCores()
```

```
## [1] 4
```