ChIPXpress: enhanced ChIP-seq and ChIP-chip target gene identification using publicly available gene expression data

George Wu, Hongkai Ji

October 14, 2013

1 Introduction

ChIPx (i.e., ChIP-seq and ChIP-chip) is increasingly used to map genome-wide transcription factor (TF) binding sites. A single ChIPx experiment can identify thousands of TF bound genes, but typically only a fraction of these genes are functional targets that respond transcriptionally to perturbations of TF expression. Unfortunately, many ChIPx experiments do not have accompanying gene expression data from TF perturbation experiments, making it challenging to identify promising functional target genes for follow-up studies. ChIPXpress is a new Bioconductor package that addresses this issue by integrating ChIPx data with 20,000+ publicly available human and mouse gene expression samples. This tool can significantly improve TF target gene ranking based only on ChIPx data and increase the chances of finding real functional targets among the top ranked genes.

2 Overview

The purpose of ChIPXpress is to identify functional target genes of a TF-of-interest by ranking TF bound target genes from ChIPx data using publicly available gene epxression data. In order for ChIPXpress to accomplish this task, the user will need to analyze their own ChIPx data apriori to obtain a list of predicted TF bound genes. Then given the list of predicted TF bound genes and a database of gene expression profiles (for the same species), ChIPXpress will output a ranked list of TF target genes, where the order is determined by how likely each gene is to be an actual functional TF target. The primary function in this package that does this is called *ChIPXpress*.

The ChIPXpress function has three input arguments:

(1) `TFID`, Entrez GeneID of the TF-of-interest

(2) `ChIP`, Ranked vector of TF bound genes predicted from ChIPx data from the most to least likely to be bound by the TF.

The TF bound genes should be inputted as a character vector in Entrez GeneID format. Users will be required to process their own ChIPx data using any software of their preference to determine the initial TF bound gene rankings. To obtain this

initial input ranking from ChIPx data, we suggest using the peak rankings, which are based on the size of the detected peaks, that are typically reported by most ChIPx peak callers. Then by assigning each peak to a gene, and sorted by the highest ranked peak for each gene, one can obtain a ranked list of genes predicted to be bound by the TF.

(3) `DB`, Database of gene expression profiles. This is a processed and normalized matrix of gene expression values where the rows correspond to unqiue Gene IDs and the columns correspond to samples.

There are currently two databases - one for mouse (DB_GPL1261) and one for human (DB_GPL570) - already built and ready for input. To use them please download and install the *ChIPXpressData* package. The databases are in big.matrix format, which is a more efficient way of storing and managing large data. Each database contains thousands of gene expression profiles collected from a diverse assortment of diseases, cell types, and conditions, which provides regulatory information that helps to infer whether a TF bound gene is a functional target. Later on, we will show how to load either of the two databases in a example ChIPXpress analysis.

Alternatively, users can build their own gene expression databases from NCBI GEO by using functions *buildDatabase* and *cleanDatabase*. This is for users who wish to analyze a different platform or species besides mouse or human, or if a user simply wants to generate their own database. The details will be explained in a later section.

3 ChIPXpress Example

Here, we illustrate an example of how to use the ChIPXpress function to produce functional TF target gene rankings. Suppose we are interested in studying Oct4 regulation in mouse embryonic stem cells (ESCs). First, we process the ChIP-seq data using CisGenome (or other method) to obtain a list of predicted Oct4-bound target genes in ESCs. This has already been done previously and is stored as a data frame in the package ready for input into the ChIPXpress function.

```
> library(ChIPXpress)
> data(Oct4ESC_ChIPgenes)
> head(Oct4ESC_ChIPgenes)
```

```
  Rank   Chr     Start       End Strand    Annotation      Gene EntrezID
1    2  chr9  50910615  50911874      + 4833427G06Rik NM_177702   235345
2    5 chr11  69394955  69396249      +         Trp53 NM_011640    22059
3    6  chr2  51926560  51928414      +          Rif1 NM_175238    51869
4   13 chr11 102190585 102191494      +          Ubtf NM_011551    21429
5   28 chr19  30103185  30104129      +         Uhrf2 NM_144873   109113
6   29  chr7 126135800 126136604      +        Gprc5b NM_022420    64297
  peak_length FDR left_peakboundary right_peakboundary peak_summit bound_center
1        1260   0          50911137           50911247    50911212     50911192
2        1295   0          69395492           69395557    69395487     69395524
3        1855   0          51927707           51927817    51927772     51927762
4         910   0         102190977          102191077   102191007    102191027
5         945   0          30103577           30103737    30103662     30103657
6         805   0         126136142          126136247   126136217    126136194
  bound_width      maxT  maxT_pos max_log2FC maxFC_pos minuslog10_minPoisP
```

```
1         111 11.045274  50911052   7.348307  50911052                 100
2          66 10.712434  69395637   7.124972  69395637                 100
3         111 10.667485  51927627   7.094812  51927627                 100
4         101 10.195434 102191197   6.778066 102191197                 100
5         161  9.808593  30103622   6.518497  30103622                 100
6         106  9.723948 126136167   6.461699 126136167                 100
  minPoisP_pos
1     50911017
2     69395392
3     51926997
4    102190952
5     30103552
6    126136097
```

Next, we need to load the pre-built mouse database of gene expression profiles from the GPL1261 platform by loading the *ChIPXpressData* package. Remember, since the database is stored in big.matrix format, we need to use the functions specially designed to work with big.matrixes. This requires installing and loading the *bigmemory* package.

```
> library(ChIPXpressData)
> library(bigmemory)
> path <- system.file("extdata",  package="ChIPXpressData")
> DB_GPL1261 <- attach.big.matrix("DB_GPL1261.bigmemory.desc", path=path)
```

To be more clear on exactly what we just did, we first located the path in which the DB_GPL1261 database is stored - which would be in the extdata folder of the installed ChIPXpressData package - and then specified the file name and the path to load the DB_GPL1261 database. To load the DB_GPL570 database for human data, we would simply replace DB_GPL1261 with DB_GPL570.

We are now ready to run the ChIPXpress function. We specify the Entrez GeneID of the TF-of-interest (18999 is the Entrez GeneID of Oct4), the vector of EntrezIDs for the predicted Oct4 bound genes, and the database:

```
> Output <- ChIPXpress(TFID="18999",ChIP=Oct4ESC_ChIPgenes$EntrezID,DB=DB_GPL1261)
> head(Output[[1]])

 18999  17865 381591  22702  22271  99377
   5.3    6.1   15.8   20.8   22.0   26.0

> head(Output[[2]])

[1] 338369 238555 257963 242860 212569 243881
```

The output is a list of size two. The first item in the list is the Oct4 target gene rankings, where the names of the vector correspond to the Entrez GeneID of each gene and each individual value is ChIPXpress score for the gene calculating by combining the ChIPx and truncated absolute correlation rankings. The second item reports the TF bound genes that were not found in the database (i.e. not measured by the microarray platform).

For the final step, you can convert the Output into a clean table with genes names or any other preferred gene identifier by using any of your favorite annotation packages (e.g., biomaRt). Here, we can use the original Oct4ESC_ChIPgenes dataframe to do so directly.

```
> GeneNames <- Oct4ESC_ChIPgenes$Annotation[match(names(Output[[1]]),
+                                                 Oct4ESC_ChIPgenes$EntrezID)]
> Result <- data.frame(1:length(Output[[1]]),GeneNames,names(Output[[1]]),Output[[1]])
> colnames(Result) <- c("Rank","GeneNames","EntrezID","ChIPXpressScore")
> head(Result)
```

	Rank	GeneNames	EntrezID	ChIPXpressScore
18999	1	Pou5f1	18999	5.3
17865	2	Mybl2	17865	6.1
381591	3	L1td1	381591	15.8
22702	4	Zfp42	22702	20.8
22271	5	Upp1	22271	22.0
99377	6	Sall4	99377	26.0

To read about exactly how (and why) ChIPXpress uses the gene expression databases, refer to the ChIPXpress paper (Wu and Ji, 2012).

An additional option for the pre-built databases is to specify the variance of the probes prior to standardization, then ChIPXpress wil check to see if the TF has a low variance expression in the database (see the ChIPXpress R help file for more details).

4 Building your own database

In some cases, the user may not want to use either of the two pre-built databases. This may occur if you would like to make your own database for a different platform, or you would like to obtain ChIPXpress rankings for a different species.

We will first go over the overall process of how the pre-built DB_GPL1261 and DB_GPL570 database are constructed and how to build your own database:
(1) download the samples of interest from NCBI GEO,
(2) process them using frma,
(3) convert array IDs into Entrez GeneIDs,
(4) modify the database such that each EntrezGeneID is in one-to-one correspondence with a row of expression measuremnts,
(5) normalize the database by rows.

Downloading and processing all samples files for a given platform will take an extremely long time due to the sheer number of samples available in NCBI GEO. For example, building a database from all the GPL1261 samples currently available in GEO (29000+ samples) would take up to 2 weeks depending on download speed and processing power. Thus, we highly recommend the user to use the pre-built databases UNLESS you absolutely need to build your own database.

Fortunately, we provide functions to do accomplish the above 5 steps in 3 steps:

(1-2) Run the *buildDatabase* function. *buildDatabase* will download the samples from NCBI GEO using the *GEOquery* package and process them with frma, by using the *frma* and *affy* packages. The output will be a matrix of expression values in big.matrix format where each row corresponds to a probe on the platform and each column corresponds to a sample.

For example, if we want to build a database from all samples files in NCBI GEO for the GPL1261 platform, we would do the following:

```
> library(bigmemory)
> library(biganalytics)
> library(ChIPXrpess)
> library(GEOquery)
> library(affy)
> library(frma)
> library(mouse4302frmavecs)
> DB <- buildDatabase(GPL_id='GPL1261',SaveDir=tempdir())
> ## Make sure the save directory is already created and empty!
```

Alternatively, we can specify specific sample files in NCBI GEO using GSM IDs. The GSM files need to all be from the same platform. Only use one of the input methods, if GSMfiles are specified then GPL_id is not required, and vice versa.

```
> GSM_ids <- c("GSM24056","GSM24058","GSM24060","GSM24061",
+              "GSM94856","GSM94857","GSM94858","GSM94859")
> DB <- buildDatabase(GSMfiles=GSM_ids,SaveDir=tempdir())
```

As you can see, this function requires the use of the *GEOquery*, *affy*, and *frma* packages to download and process the data and requires the use of the *bigmemory* and *biganalytics* packages to work with big.matrix format files. We also need to load the *mouse4302frmavecs* package, since it is required by frma to process the GPL1261 raw CEL files. (Note: GPL1261 is the GPL ID for mouse 430 2.0 array files) When you build your own database, you will also need to install and download the appropriate frmavecs package. If a current frmavec package does not exist for your desired platform or species, then please see the *frmaTools* package on how to construct your own frmavecs file required by frma.

(3) Convert the rownames of the matrix, currently as probeIDs specific to the platform, into Entrez GeneIDs. This step must be done by the user themselves! Feel free to use any annotation package or method to convert the probeID into Entrez GeneIDs. Here is example code of how we would do it:

```
> library(mouse4302.db)
> EntrezID <- mget(as.character(rownames(DB)),mouse4302ENTREZID)
> rownames(DB) <- as.character(EntrezID)
```

To be explicit, we first downloaded the *mouse4302.db* package which gives us the annotation information on which probeID corresponds to which Entrez GeneID for the mouse 430 2.0 array (or GPL1261 platform). Then we use mget to obtain the Entrez GeneIDs and replace the existing rownames with the Entrez GeneIDs.

(4-5) Run the *cleanDatabase* function on the annotated database. *cleanDatabase* will search for Entrez GeneIDs that corresponds to multiple rows in the database and retain only the row with the highest variance. This is to ensure a 1-to-1 correspondance between each gene ID and each row of expression measurements. Afterwards, the function will normalize the data by rows.

```
> cleanDB <- cleanDatabase(DB,SaveFile="newDB_GPL1261.bigmemory",
+                          SavePath=tempdir())
> head(cleanDB)
> ### cleanDB contains the finished database ready to be
> ### inputted into ChIPXpress.
```

The user can then directly proceed to use cleanDB as input into the ChIPXpress function. Be sure your final database includes a lot of samples in order to obtain a reliable estimate of the correlation between the TF and each gene. This is also to ensure that the regulatory information obtained from the database is based on the regulatory behavior of the TF and its target genes across a diverse assortment of cell types.

```
> out <- ChIPXpress(TFID="18999",ChIP=Oct4ESC_ChIPgenes,DB=cleanDB)
> head(out[[1]])
> head(out[[2]])
```

Or for future R sessions, the user can load the newly created database by:

```
> cleanDB <- attach.big.matrix("newDB_GPL1261.bigmemory.desc",path=tempdir())
> head(cleanDB)
```

Thus, to load the a big.matrix file, such as the pre-built database or the newly created database in this example, we simply need to specify the directory that contains the .bigmemory and .bigmemory.desc files. The .bigmemory file contains the actual matrix information and the XXX.bigmemory.desc file is the description file for the matrix. When saving new databases, feel free to name the database as you see fit, but remember to add on a XXX.bigmemory to denote that it is a big.matrix file. For more information on the big.matrix format, see the *bigmemory* help files.

One final note, it is possible to convert the probeIDs into an alternative geneID format. To do this, simply annotate the probeIDs into the alternative gene ID format and store as the rownames of the matrix. Then, when running ChIPXpress, just make sure that the inputted TF bound genes from ChIPx data are also in the same geneID format. Then ChIPXpress will output rankings using the alternative geneID format, rather than the default and preferred Entrez GeneID format.

References

[1] Barrett T., Troup D.B., Whilhite S.E., et al. (2007) NCBI GEO: mining tens of millions of expression profiles – database and tools update. *Nucl. Acids Res.*, **35**, D760–D765.

[2] McCall M.N., Bolstad B.M., and Irizarry R.A. (2010) Frozen robust multiarray analysis (fRMA). *Biostatistics*, **11**, 242–253.

[3] Wu, G. and Ji, H. (2012) ChIPXpress: enhanced ChIP-seq and ChIP-chip target gene identifi-cation using publicly available gene expression data. *In preparation.*