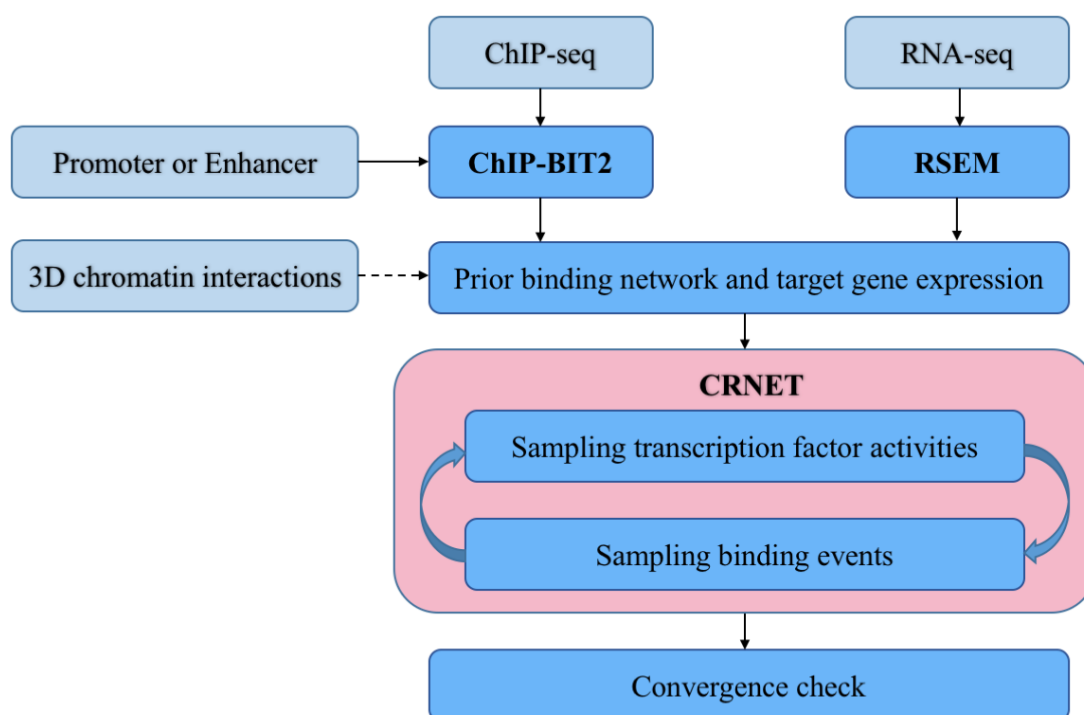


# CRNET User Manual (V2.2)

CRNET is designed to use time-course RNA-seq data for the refinement of functional regulatory networks (FRNs) from initial candidate networks that can be constructed from ChIP-seq data. CRNET uses a two-stage Gibbs sampling framework to iteratively estimate the hidden transcription factor activities and the posterior probabilities of binding events. By using a t-statistic jointly considering regulation strength and regression error, the sampling process of CRNET converges much faster than current Bayesian based regulatory network inference methods and the performance of CRNET is more robust against the noise in prior binding information and gene expression. **Fig. 1** shows the flowchart of CRNET for functional regulatory network inference. The R scripts of CRNET have been tested using R 3.3 under MAC OS 10.11 and Ubuntu 12.04 64 bit.



**Fig. S1.** A workflow of CRNET.

## 1. CRNET input data

### 1.1 FRN inference at gene promoter regions

An initial network and a time-course gene expression dataset are needed to run CRNET, which are usually the minimal requirements for functional network inference with most of the methods. CRNET accepts either weighted (0~1) or binary prior binding information. If ChIP-seq data are used, a weighted binding network can be generated using our previously developed tool ChIP-BIT2 (<http://www.cbil.ece.vt.edu/software.htm>). From other resources (such as RegNetwork (<http://www.regnetworkweb.org/>)), a binary binding network can be constructed by users as a prior. In the prior binding file, each row represents a gene and each column represents a TF.

### Load prior TF binding network at promoters

```
B<-as.matrix(read.table('TF_promoter_binding.txt', row.names = 1, header=TRUE))
TF_symbols<-colnames(B)
Gene_symbols<-row.names(B)
```

The format of prior binding matrix is defined as follows:

Gene_symbols	CEBPB	cFOS	cJUN	CREB1	...
ABAT	0.66319	0.97672	0	0.89466	
ABCE1	0	0.84328	0	0.72576	
ABHD2	0	0	0.7332	0.81202	
ACAT2	0.74986	0	0	0.7724	
...					

The first row denotes TF symbols and the first column denotes gene symbols. The prior binding information can be weighted or just '0' or '1', which can be specified using parameter 'prior\_flag'.

```
#set binary flag of prior binding matrix: 1 weighted; 0 binary
prior_flag=1
```

### Load time-course gene expression data

A time-course RNA-seq gene expression dataset is needed. We recommend using the TPM value of each gene as gene expression, which can be estimated using RSEM (<https://deweylab.github.io/RSEM/>). However, if RNA-seq data are not available, properly normalized time-course microarray gene expression data also works.

```
Gene_EXP<-as.matrix(read.table('Time_course_gene_expression.txt', row.names = 1,
header=TRUE))
```

The format of time-course gene expression data is defined as follows:

SampleID	Sample_1	Sample_2	Sample_3	Sample_4	...
Timepoints	5	10	20	40	
ABAT	-0.07744	0.053572	-0.055054	0.015113	
ABCE1	-0.13893	-0.01361	-0.09728	0.003758	
ABHD2	-0.11876	0.063178	0.02229	0.032758	
ACAT2	-0.034728	0.059747	0.13875	-0.045705	
...					

The first column of gene expression data denotes gene symbols, which must be consistent to the first column of gene symbols in the prior binding matrix. The first row denotes unique sample ID for gene expression samples and the second row is a numerical row including time points for individual gene expression samples. We provide an option to estimate the same TF activity for multiple replicates of the same time point by setting parameter 'replicate\_flag'. If only one sample is provided for each time point, as shown above, 'replicate\_flag' is set to '0'.

```
#set the flag for sample replicates under each time point: 0 independent or no-
replicates; 1 related replicates
replicate_flag=0
```

Otherwise, if two or more replicates are generated under each time point as follows:

SampleID	Sample_1_rep1	Sample_1_rep2	Sample_2_rep1	Sample_2_rep2	...
Timepoints	5	5	10	10	
...					

At the first time point, we need put the same number, 5 (5mins), in the second row for two replicates. And set 'replicate\_flag' is set to '1'.

## 1.2 FRN inference at enhancer regions

To infer FRNs at enhancer regions, we will need a prior binding matrix containing distal binding events at enhancer regions and another enhancer-gene map to associate those distal binding events with target genes. ChIP-BIT2 can also be used to detect binding events at enhancer regions. Alternatively, users can generate their own binding profiles at enhancer regions, which can be either weighted or binary.

### *Load TF binding network at enhancers*

```
B<-as.matrix(read.table('TF_enhancer_binding.txt', row.names = 1, header=TRUE))
TF_symbols<-colnames(B)
prior_flag=1
```

Enhancer_ID	CEBPB	cFOS	cJUN	CREB1	...
Distal-Prediction-162	0.81353	0	0	0.56975	
Distal-Prediction-169	0	0	0	0.59993	
Distal-Prediction-358	0.64626	0	0	0.82118	
Distal-Prediction-464	0	0	0	0.64376	
Distal-Prediction-671	0	0	0	0	
...					

Here, the first row still represents individual TFs but the first column denotes enhancer ID instead of gene symbols since each TF binds to enhancer regions directly and indirectly regulate enhancer target gene expression.

### *Load enhancer-gene loop map*

```
Enhancer_gene_loop<-as.matrix(read.table('Enhancer_Gene_map.txt', row.names = 1,
header=TRUE))
```

Enhancer_ID	CELSR2	CTPS1	LHX4	RAP1A	...
Distal-Prediction-162	0	0	1	0	
Distal-Prediction-169	0	0	1	0	
Distal-Prediction-358	0	1	0	0	
Distal-Prediction-464	1	0	0	0	
Distal-Prediction-671	0	0	0	1	
...					

The enhancer-gene loop input is the key to link binding events at enhancer regions to target genes. The first column represents enhancer ID, which must be consistent with prior binding matrix. The first row denotes gene symbols, which needs to be consistent with gene symbols in the gene expression data.

### *Load time-course gene expression data*

```
Gene_EXP<-as.matrix(read.table('Enhancer_target_gene_expression.txt', row.names = 1,
header=TRUE))
replicate_flag=0
```

SampleID	Sample_1	Sample_2	Sample_3	Sample_4	...
Timepoints	5	10	20	40	
CELSR2	-0.1137	0.11911	0.1428	0.17023	
CTPS1	-0.4027	-0.33153	0.014567	-0.28826	
LHX4	-0.074116	-0.10675	-0.18672	0.03247	
RAP1A	-0.19733	-0.040421	0.21822	0.16403	

...  
...

## 2. CRNET workflow

### 2.1 Hyper-parameters

As introduced in the CRNET paper, two hyper-parameters are needed as gene expression data noise variance (sigma\_noise) and transcription factor activity (TFA) noise variance (sigma\_X). In some methods these parameters may be assumed as random variables and sampled together with the other major variables under a Bayesian framework. However, the distribution of these variance variables are hard to know and there is no clear evidence to show that sampling these parameters can really improve the overall performance. Therefore, to make the sampling process more efficient, we directly set them as hyper-parameters with fixed values.

In detail, we set the value of 'sigma\_noise' to 1, as informative prior control over the noise in gene expression data because RNA-seq data are used in this study. Different value settings of sigma\_X from 1 to 100 has been discussed in BNCA (Sabatti and James, 2006). Network prediction performances are quite similar using different values and the author suggested using an informative prior as 1 to speed up the convergence. Definitely, these two parameters can be adjusted according to the gene expression data quality.

```
# CRNET model hyper-parameters
sigma_noise=1 # Gene expression data noise variance
sigma_X=1 #TFA variance
```

### 2.2 CRNET two stage sampling

CRNET iteratively samples the binding network and transcription factor activities using a Gibbs sampling framework. In this demo case, we set the total number of iterations to 1000. We provide two different functions to infer FRNs, respectively, at promoter and enhancer regions.

```

#FRN inference at promoter region
Sampling_results<-CRNET_promoter(Gene_EXP, B, prior_flag, sigma_X, sigma_noise, b1, b0,
Num_iteration, replicate_flag)

#FRN inference at enhancer region
Sampling_results<-CRNET_enhancer(Gene_EXP, Enhancer_gene_loop, B, prior_flag, sigma_X,
sigma_noise, b1, b0, Num_iteration, replicate_flag)

```

## 2.3 CRNET logistic function parameter training (optional)

We define a logistic function to convert the t-statistic value for each binding into a probability. The training procedure of logistic function parameters is summarized as follows:

(1) Randomly select a number of TFs (smaller than that of gene expression samples) and run Network Component Analysis (NCA) to estimate hidden TFA and regulation strength;

```

TF_index = randsample(T,M-1);
A_selected=A(:, TF_index);
% Network Component Analysis
[Ae,Se] = FastNCA(Y,A_selected);

```

(2) For each binding event, covert regulation strength and regression error into to a t-score and make a local judgement as '1' if the t-score is larger the value with a false positive rate < 0.05; otherwise as '0';

```

for g=1:G
    ssr=sum((Y(g,:)-Ae(g,:)*Se).^2);
    for t=1:size(A_selected,2)
        if A_selected(g,t)>0
            C=1/(sum(Se(t,:).^2));
            t_statistics(g,( t)=Ae(g,t)/sqrt(C*ssr/(M-1-sum(A_selected(g,:))));
            threshold=tinv(0.975,M-1-sum(A_selected(g,:)));
            %Logistic judgement
            if abs(t_statistics(g, t))>=threshold
                logistic_flag(g,( t)=1;
            end
        end
    end
end
end

```

(3) Repeat (1) and (2) 100 times to test all TFs sufficiently;

(4) Run a logistic regression on all t-scores as well as their labels. We use a MATLAB implementation of NCA (<http://www.eee.hku.hk/~cqchang/FastNCA.htm>) and provide a short MATLAB script 'CRNET\_logistic\_function\_training.m' for parameter training. Using a MATLAB function of glmfit, we can perform logistic regression by setting function parameters as 'binomial', 'logit'.

```

%Logistic regression
label=logistic_flag(A_selected>0);
B = glmfit(abs(t_statistics(A_selected>0)), [label ones(size(label))], 'binomial',
'logit');

```

Values of b1 and b0 can be found in the output vector B = [b0, b1].

### 3 CRNET output

CRNET will output a matrix to denote the predicted FRN at promoter regions where each row represents a gene and each column represents a proximal TF.

```
colnames(Sampling_results$Zf) <- TF_symbols
rownames(Sampling_results$Zf) <- Gene_symbols
write.csv(Sampling_results$Zf, file = 'Promoter_FRN.csv', quote = FALSE)
```

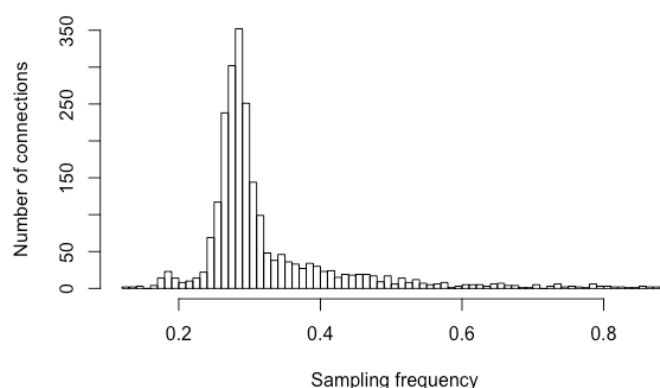
Gene_symbols	CEBPB	cFOS	cJUN	CREB1	...
ABAT	215	225	0	196	
ABCE1	0	214	0	201	
ABHD2	0	0	206	186	
ACAT2	195	0	0	228	
...					

In the predicted FRN at enhancer regions, each row represents an enhancer-gene interaction as enhancer\_ID:gene\_ID and each column represents a distal TF.

```
colnames(Sampling_results$Zf) <- (TF_symbols)
rownames(Sampling_results$Zf) <-
paste(Sampling_results$Enhancer_ID[Sampling_results$Enhancer_gene_loops[,1]],
Sampling_results$Gene_symbols[Sampling_results$Enhancer_gene_loops[,2]], sep=":")
write.csv(Sampling_results$Zf, file = 'Enhancer_FRN.csv', quote = FALSE)
```

	CEBPB	cFOS	cJUN	CREB1	...
Distal-Prediction-162: LHX4	235	0	0	225	
Distal-Prediction-169: LHX4	0	0	0	211	
Distal-Prediction-358: CTPS1	223	0	0	214	
Distal-Prediction-464: CELSR2	0	0	0	211	
Distal-Prediction-671: RAP1A	0	0	0	0	
...					

We calculate a sampling frequency for each unit as Num. of samples/Total rounds of sampling. We recommend users set the cut-off threshold after examining the overall distribution of sampling frequencies as shown in the following figure (**Fig. 2**). Binding network density, number of TFs, number of genes, gene expression data quality and number of expression samples may all affect the learned distribution of binding connections.



**Figure 2. Distribution of samples on all binding connections (demo).**

## Reference

Sabatti, C. and James, G.M. Bayesian sparse hidden components analysis for transcription regulation networks. *Bioinformatics* 2006;22(6):739-746.