CRNET User Manual (V2.0)

We develop an efficient Bayesian integration method, namely CRNET, for functional regulatory network inference using 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.

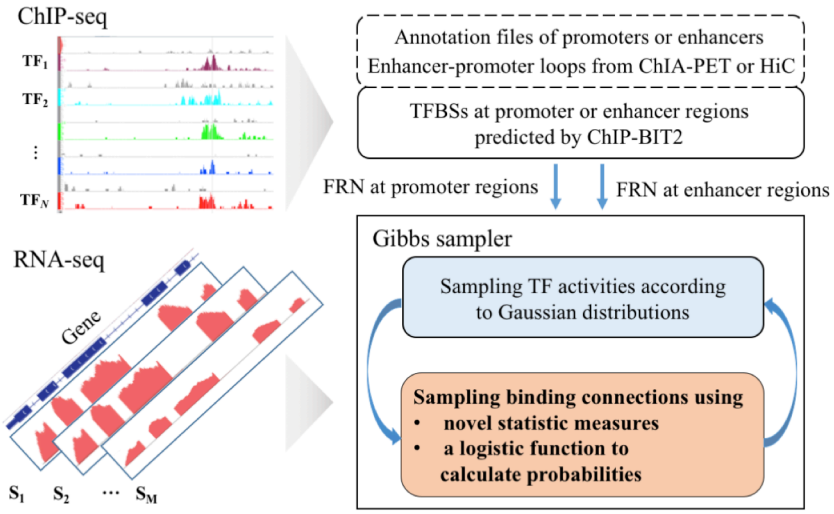

Figure 1. Flowchart of CRNET for FRN inference. CRNET is built on a two-stage Gibbs sampling procedure: (1) sampling hidden transcription factor activities (TFAs) and (2) sampling binding connections.

1. CRNET input data

FRN inference at gene promoter region

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 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)
```

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

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 work. In the input file, the second row should contain the information of time points. We provide an option to estimate the same TF activity for multiple replicates of the same time point. They can be also treated as independent samples by setting the "`replicate_flag`".

```
# load time-course gene expression data Y
# the second row represents time points
Gene_EXP<-as.matrix(read.table('Time_course_gene_expression.txt', row.names = 1,
header=TRUE))

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

FRN inference at gene promoter region

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)

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

# load enhancer gene loop map
Enhancer_gene_loop<-as.matrix(read.table('Enhancer_Gene_map.txt', row.names = 1,
header=TRUE))

# load gene expression data Y
Gene_EXP<-as.matrix(read.table('Enhancer_target_gene_expression.txt', row.names = 1,
header=TRUE))

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

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. 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
```

(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');
```

3 CRNET output

In the predicted FRN at a promoter region, 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)
```

In the predicted FRN at an enhancer region, 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)
```

The output `Sampling_results$Zf` of CRNET is a weighted binding network where the number of samples on each binding represents the posterior probability. We recommend users set the cut-off threshold after examining the overall distribution of Gibbs samples on all bindings 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.

```
hist(Sampling_results$Zf[which(Sampling_results$Zf>0)], breaks=100, xlab = 'Sampling
frequency', ylab = 'Number of connections')
```

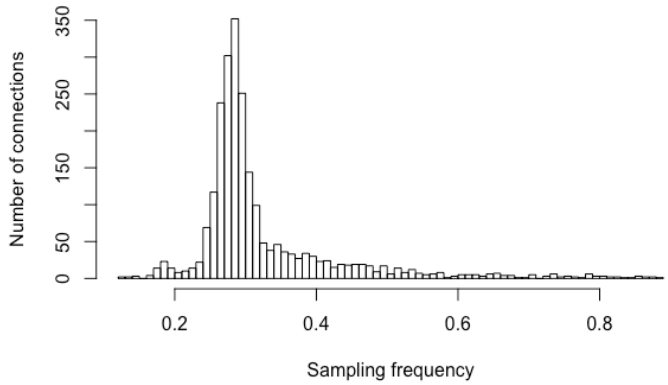

Figure 2. Distribution of samples on all binding connections (demo).