BayesCRE package:

The R package BayesCRE, consists of several function that perform various tasks ranging from pre-processing of the network and data, generation of input to the Bayesian network, construction of the Bayesian network and the Gibbs sampling inference. The details of the algorithm can be found in:

"Molecular causes of transcriptional response: A Bayesian prior knowledge approach", Kourosh Zarringhalam, Ahmed Enayetallah, Alex Gutteridge, Ben Sidders, Daniel Ziemek, Bioinformatics 2013.

BayesCRE Directory includes the following:

A) data folder: This folder contains the following files:

- **a.** Free open source small scale Knowledge Base (KB) from OpenBEL portal (www.openbel.org) under a Creative Commons license. Please check the documents at openbel.org. The folder KB contains:
  - **i.** BEL_LargeCorpus.ents: KB entries.
  - **ii.** BEL_LargeCorpus.rels: KB causal relations.
  - **iii.** MeSH.txt: This file contains the PMIDS and associated MeSH terms from PubMed for the included KB. Note, if you are using your own KB, you should provide MeSH terms as well.

- **b.** Simulations: Contains simulated data file using the above KB. The included simulation was generated by assuming that the regulators **HRAS**, **FOXO3** and **KLF1** are **up**, **down** and **up** regulated respectively.

B) algorithms.R: Contains the algorithms for running the inference on the KB.

C) BayesCRE.R: An executable R script that runs the appropriate functions in the algorithm.R

D) simulateData.R: An R script that simulates expression data given a causal network.

E) run_simulation.sh: A sample command line run script.

F) output: The outputs of the algorithm are placed here. In particular "sim_hyp.txt" contains the hypothesis recovered by the algorithm that contains all the hypotheses used to simulate the data.

Running the main inference program:

You need R (cran.org) and the following libraries installed: 'plyr', 'optparse', 'fdrtool'. To get help on command line arguments type: ./BayesCRE -h

To run the code on the included simulated data, please use the provided shell script as: ./run_simulations.sh

To simulate other data sets, please see: simulateData.R

Description of the functions in the package:

1. **BayesCRE.R**
   This is the main "run" file that calls the appropriate functions. It reads in the input files and the parameters. The list of the input arguments is documented in the code. Next it processes and prunes the network so it consists of Proteins, mRNAs and compounds only. The depth of the network is 1. The top layer consists of Proteins and compounds (upstream regulators) and the bottom layer are the genes (mRNAs). From this processed network, a Causal network is constructed as follows. For each gene, a hidden (explicit) node is added that models the true state of the gene (to account for the error in gene expression data). For each edge, the pubmed id and the associated MeSH terms are identified and an applicability node is added to the network. The applicability nodes are directly connected to the gene of the corresponding edge. The parents of the applicability nodes are the MeSH terms (context nodes) that the article reporting the edge is annotated with. The number of context nodes to be included is limited to those that are 1) enriched according to the gene expression data (--cutoffq) and 2) are not connected to more than a pre-specified number of edges (-mn). Moreover, the prior probability of the edge being in context is also an input parameter and depends on the value of the gene (--pc and --pa). Other parameters include the prior probabilities for regulator nodes (--hz), prior probability of the context nodes (--cp), false positive rate of the gene expression data (--alpha) and false negative rate of the gene expression data (--beta). For a complete list of the input arguments see the core (./BayesCRE.R --help).

   Once the data and the network are processed the Gibbs sampler is called to perform the inference and calculate the marginal probabilities of the upstream nodes.

2. **Algorithms.R**
   This file contains the main algorithms.

a. **processKB:** Reads in and cleans up the network. Removes unneeded columns
b. **genOneLevel:** Generates the network of Proteins, mRNAs, and Compounds. Proteins and compounds are directly connected to the genes. Protein-Protein interactions are not taken into consideration.
c. **genContextGraph:** Reads in the input gene expression data and integrates the applicability nodes and context nodes. To reduce the number of MeSH terms that are integrated in the network, an enrichment analysis is performed to include only the MeSH terms that are enriched according to genes whose values are altered (i.e., non-zero genes). Among the enriched MeSH terms, only those that are connected to no more than mn terms are considered. The algorithm outputs a table with rows consisting of the source, the target, pmid, and MeSH term. Rows with identical source and target will be repeated according to the number of MeSH terms.

3. **gibbsSampler:**
   This is the core of the algorithm and the most time consuming part. A vector X is created that contains the values of all nodes. Each node in the network will have an index in the vector X (arbitrary). For easy lookup of the parents and children of the nodes, lookup tables "relsBySrcUid", "relsByTrgUid", "relsByMeSHId", "relsByAppId" are created that hold the proper indices of parents and children of the nodes as sorted in the vector X. The algorithm proceeds by initializing the vector X and iteratively updating the value of the nodes in X according to the conditional probability tables in the Markov blanket of the node. Depending on the type of the node, the conditional probabilities are defined in the paper. In each pass in the for-loop, the values of one upstream node (Proteins, Compounds), one applicability node and one context node will be updated. Since there are many upstream nodes (>10,000) and since the Markov blanket of these nodes can be very large, these conditional probability tables are not stored in the memory and are calculated each time in the for loop. Note that when updating the values of upstream nodes, only the in-context network is used, i.e., the part of the network whose applicabilities are non-zero. After a burn in period (--burnin), the sampled values in X for each node (apart from the evidence nodes, i.e., mRNAs) are added to compute the marginal probability. The algorithm will terminate after the total number of samples (--samples) have been exhausted.

4. There are several other auxiliary functions in the code such as "comb", "compProbH" and "pZ" that compute the combinatorial parts of the conditional probabilities as explained in the paper. Other functions such as "addToChain", and "which_central" are used for optimization reasons. Additionally, there are functions that slice the network and collect some statistics on the nodes. For example "nodeNet", takes the id of a node and returns the part of the network that involves the node. "nodeNetList" is similar to "nodeNet" but takes a list of ids rather than one id. "node.stat"

computes the number of correct and incorrect predictions made by a regulator node. “node.stat.list” performs the same function, but for a list of regulator nodes.

The code is commented and the method is fully described in the paper.