XMRF: An R Package to Fit Markov Networks to High-Throughput Genomics Data

Ying-Wooi Wan, Genevera I. Allen, Yulia Baker,
Eunho Yang, Pradeep Ravikumar, Zhandong Liu

June 29, 2015

Contents

1 Introduction

XMRF is an R package implemented to enable biomedical researchers to discover complex interaction between genes from multi-dimensional genomics data. These interactions are mathematically captured in the form of Markov Networks. Methods implemented in the package support data produced by different high-throughput genomic profiling technologies including gene expression from DNA microarray, genomic mutation from SNP array, copy number variation from array-CGH, DNA methylation profiles from methylation array, and read counts from next-generation sequencing like RNA-Seq.

This package encodes the recently proposed parametric family of graphical models based on node-conditional univariate exponential family distributions (Yang *et al.*, 2012, 2013a). Specifically, our package has methods for estimating Gaussian graphical model (Meinshausen and Buhlmann, 2006), Ising model (Ravikumar *et al.*, 2010), and the Poisson family graphical models (Allen and Liu, 2012, 2013; Yang *et al.*, 2013b). This allows the user to choose the right graphical model according to the native distribution of the genomics data.

To identify the optimal network sparsity, this package implements two data-driven approaches to select the sparsity of a fitted network: a stabilty-based approach for a single regularization value over many bootstrap resamples (Meinshausen and Bühlmann, 2010), or an averaged stability-based approach computed over a range of regularization values via StARS (Liu *et al.*, 2010).

The graph estimation techniques implemented in this package follow the neighborhood selection graph estimation technique by proximal or projected gradient descent using warm starts over the range of regularization parameters, which allows for estimation of the neighborhood for each node independently (Meinshausen and Buhlmann, 2006; Liu *et al.*, 2010). Due to the parallel nature of this algorithm, high computational efficiency is achieved through implementation of forking with functions from parallel (R Core Team, 2013) and snowfall (Knaus, 2013) package. Furthermore, this package also depends on glmnet package (Friedman *et al.*, 2010) for L_1 norm regularized linear regression, igraph package (Csardi and Nepusz, 2006) for graph manipulation, and TCGA2STAT package() to import and process high-throughput genomics data.

2 Licensing, Availability and Installation

The XMRF package and the underlying code in the package are distributed under the GPL2.0 license. Users of the package are free to use and redistribute the package but editing are prohibited. The XMRF package is available from the CRAN Project and the package's Github.
To install XMRF from CRAN Project, use the GUI or install.packages command by specifying the parameters.

To install XMRF from the package archive file obtained from the package's Github :

```
> install.packages("XMRF_1.0.tar.gz", repos = NULL, type = "source")
```

3 Choosing the Right Graphical Model for Genomics Data

The development of high-throughput technology, such as microarray, SNP array, array-CGH, methylation array, exome-sequencing, and RNA-sequencing, has generated a wide variety of genetics data. To accurately estimate the underlying network structure from these data types, one needs to apply the right network inference algorithm based on the platform-specific data distribution.

In XMRF, data are modeled using their native distribution instead of normalizing the data to follow a Gaussian distribution. To accomplish this, our package implements methods for three families:

Gaussian graphical models (`GGM`), Ising models (`ISM`), and Poisson family graphical models including regular Poisson graphical models (`PGM`) as well as several variants of the Poisson family of models such as the truncated Poisson (`TPGM`), sub-linear Poisson (`SPGM`), and local Poisson (`LPGM`) (Allen and Liu, 2013). Note that (Yang *et al.*, 2013b) proposed all these variants of the Poisson family as the regular Poisson graphical model only permits negative conditional dependencies between nodes; each of these variants relaxes restrictions resulting in both positive and negative conditional dependencies. For genomic networks based on sequencing data, we recommend using the `LPGM` variant as proposed in (Allen and Liu, 2013), noting that this local model closely approximates the proper MRF distribution of the `SPGM` formulation (Yang *et al.*, 2013b). In the following table, we provide our recommendations for which distributional families to model for data from different high-throughput platforms.

Data	Data type	`XMRF` method
RNA-Sequencing	Count data	`LPGM,SPGM`
Microarray/ Methylation	Gaussian data	`GGM`
Mutation/ CNV	Binary data	`ISM`

3.1 Poisson Graphical Model for NGS data

Count data generated by next generation sequencing (NGS) is a good example of why parametric families of Markov Networks beyond Gaussian graphical models are needed. This read count data is highly skewed and has large spikes at zero so that standardization to a Gaussian distribution is impossible. Here, we demonstrate through a real example how to process NGS data so that we can use Poisson family graphical models to learn the network structure. Our processing pipeline is given in (Allen and Liu, 2012) and encoded in the `processSeq` function of the package.

The level 3 RNA-Seq (UNC Illumina HiSeq RNASeqV2) data consisting of 445 breast invasive carcinoma (BRCA) patients from the Cancer Genome Atlas (TCGA) project was obtained. The set of 353 genes with somatic mutations listed in the Catalogue of Somatic Mutations in Cancer (COSMIC) were further extracted. The data was prepared and stored as the `brcadata` data object included in the package. The values in this data object are the normalized read counts (RSEM) obtained from TCGA data download portal, representing the mRNA expression profiles of the genes. The data matrix is of dimension pxn. Figure S1 shows that the data is more appropriate for Poisson family graphical models after being preprocessed with the `processSeq` function as given in the following code snippets:

```
> library(XMRF)
> data('brcadata')
> brca = t(processSeq(t(brcadat), PercentGenes=1))
```

To estimate the underlying network structure of the count-valued data, `XMRF` implements four different models from the Poisson family graphical models: regular Poisson graphical model (`PGM`) that only permits negative conditional dependencies, truncated Poisson (`TPGM`), sub-linear Poisson (`SPGM`), and local Poisson (`LPGM`) (Allen and Liu, 2013). The latter three models are variants of the Poisson family that relax restrictions as imposed in a regular Poisson model, resulting in both positive and negative conditional dependencies (Yang *et al.*, 2013b). `TPGM` should be used if one wants to truncate the large counts observed in NGS dataset. Alternatively, `SPGM` implements a sub-linear truncation for the NGS data which gives a softer reduction on large counts. `LPGM` is a faster algorithm that approximates the Markov Network while preserving both positive and negative relationship (Allen and Liu, 2013).

In practice, we choose `LPGM` since it is the fastest and most flexible way to capture both positive and negative dependencies:

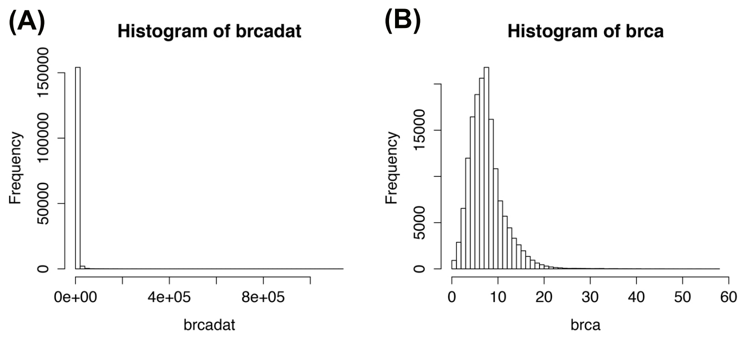


Figure S1: Distribution of TCGA BRCA RNA-Seq data before (A) and after (B) preprocessing. The latter gives a distribution more appropriate for Poisson family graphical models.

```
> p = nrow(brca)
> n = ncol(brca)
> lambda = lambdaMax(t(brca)) * sqrt(log(p)/n) * 0.1

# Run Local Poisson Graphical Model (LPGM)
> lpgm.fit <- XMRF(brca, method="LPGM", N=1, lambda.path=lambda)
```

3.2 Gaussian Graphical Model for Expression Arrays

When genomics data is profiled with microarrays, such as with mRNA arrays, miRNA arrays, or methylation arrays, Gaussian graphical models should be used to estimate the network structures. Similar workflows as presented in last section can be applied to fitting Gaussian Markov Networks to data that approximately follows a multivariate Gaussian distribution.

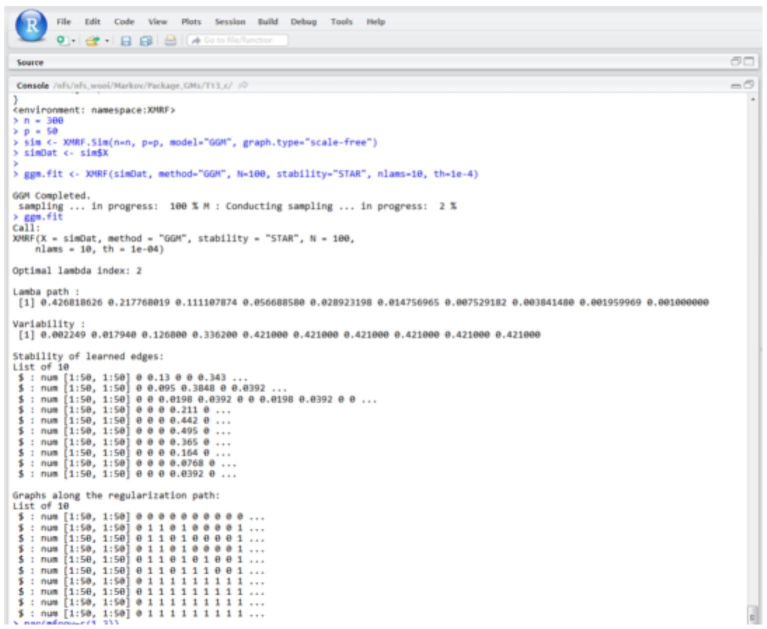
Here, we show an example on fitting simulated Gaussian multivariate data with `GGM` (Gaussian graphical model). Output of the code snippets below is shown in Figure S2.

```
> library(XMRF)
> n = 300
> p = 50

# Simulate a scale-free network from multivariate Gaussian data
> sim <- XMRF.Sim(n=n, p=p, model="GGM", graph.type="scale-free")
> simDat <- sim$X

# Run Gaussian Model and display the results
> ggm.fit <- XMRF(simDat, method="GGM", N=100, stability="STAR", nlams=10, beta=0.05, th=1e-4)
> ml = plotNet(sim$B)
> ml = plot(ggm.fit, mylayout=ml)
```

3.3 Ising Graphical Model for Mutation Data

To fit Markov Networks to binary data, the `XMRF` function with `method="ISM"` can be used. In this section, we give an example of fitting an Ising model to simulated data with a lattice graph as well

(A)



```
}
<environment: namespace:XMRF>
> n = 300
> p = 50
> sim <- XMRF.Sim(n=n, p=p, model="GGM", graph.type="scale-free")
> simDat <- sim$X
>
> ggm.fit <- XMRF(simDat, method="GGM", N=100, stability="STAR", nlams=10, th=1e-4)

GGM Completed.
 sampling ... in progress:  100 % M : Conducting sampling ... in progress:  2 %
> ggm.fit
Call:
XMRF(X = simDat, method = "GGM", stability = "STAR", N = 100,
    nlams = 10, th = 1e-04)

Optimal lambda index: 2

Lamba path :
 [1] 0.426818626 0.217768019 0.111107874 0.056688580 0.028923198 0.014756965 0.007529182 0.003841480 0.001959969 0.001000000

Variability :
 [1] 0.002249 0.017940 0.126800 0.336200 0.421000 0.421000 0.421000 0.421000 0.421000 0.421000

Stability of learned edges:
List of 10
 $ : num [1:50, 1:50] 0 0.13 0 0 0.343 ...
 $ : num [1:50, 1:50] 0 0.095 0.3848 0 0.0392 ...
 $ : num [1:50, 1:50] 0 0 0.0198 0.0392 0 0 0.0198 0.0392 0 0 ...
 $ : num [1:50, 1:50] 0 0 0 0.211 0 ...
 $ : num [1:50, 1:50] 0 0 0 0.442 0 ...
 $ : num [1:50, 1:50] 0 0 0 0.495 0 ...
 $ : num [1:50, 1:50] 0 0 0 0.365 0 ...
 $ : num [1:50, 1:50] 0 0 0 0.164 0 ...
 $ : num [1:50, 1:50] 0 0 0 0.0768 0 ...
 $ : num [1:50, 1:50] 0 0 0 0.0392 0 ...

Graphs along the regularization path:
List of 10
 $ : num [1:50, 1:50] 0 0 0 0 0 0 0 0 0 0 ...
 $ : num [1:50, 1:50] 0 1 1 0 1 0 0 0 0 1 ...
 $ : num [1:50, 1:50] 0 1 1 0 1 0 0 0 0 1 ...
 $ : num [1:50, 1:50] 0 1 1 0 1 0 0 0 0 1 ...
 $ : num [1:50, 1:50] 0 1 1 0 1 0 1 0 0 1 ...
 $ : num [1:50, 1:50] 0 1 1 0 1 1 1 0 0 1 ...
 $ : num [1:50, 1:50] 0 1 1 1 1 1 1 1 1 1 ...
 $ : num [1:50, 1:50] 0 1 1 1 1 1 1 1 1 1 ...
 $ : num [1:50, 1:50] 0 1 1 1 1 1 1 1 1 1 ...
 $ : num [1:50, 1:50] 0 1 1 1 1 1 1 1 1 1 ...
```

(B)

(C)

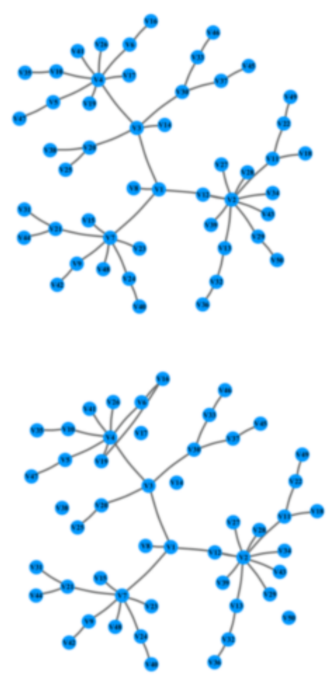


Figure S2: Results of fitting a Gaussian graphical model on simulated multivariate Guassian data. (A) Screen shot of the commands and output of fitting `GGM` model over a path of 10 regularization parameters from R studio. Simulated true network (B) and the estimated network from `XMRF` (C).

as estimating interactions among mutated genes in TCGA lung squamous cell carcinoma (LUSC) samples.

3.3.1 Learning a Lattice Graph from Simulated Data

In the following example, a multivariate binary data matrix of 400 observations that will give a 5 x 5 grid graph will be simulated. Our `XMRF` Ising model is fit to infer the lattice network from simulated data. Models were fitted over a path of 20 regularization values. StARS stability selection with 100 iterations were used to select the optimal network. Figure S3 shows the original network and estimated optimal network plotted with the `GMS`'s default `plot` function and R `image` function.

```
> n = 400
> p = 25
> simdat <- XMRF.Sim(n=n, p=p, model="ISM", graph.type="lattice")
> ismfit <- XMRF(simdat$X, method="ISM", N=100, nlams=20,
            stability="STAR", th=0.1, beta=0.1)

> par(mfrow=c(2,2))
> image(simdat$B)
> image(ismfit$network[[ismfit$opt.index]])
> ml = plotNet(simdat$B, fn="")
> ml = plot(ismfit, fn="", mylayout=ml)
```

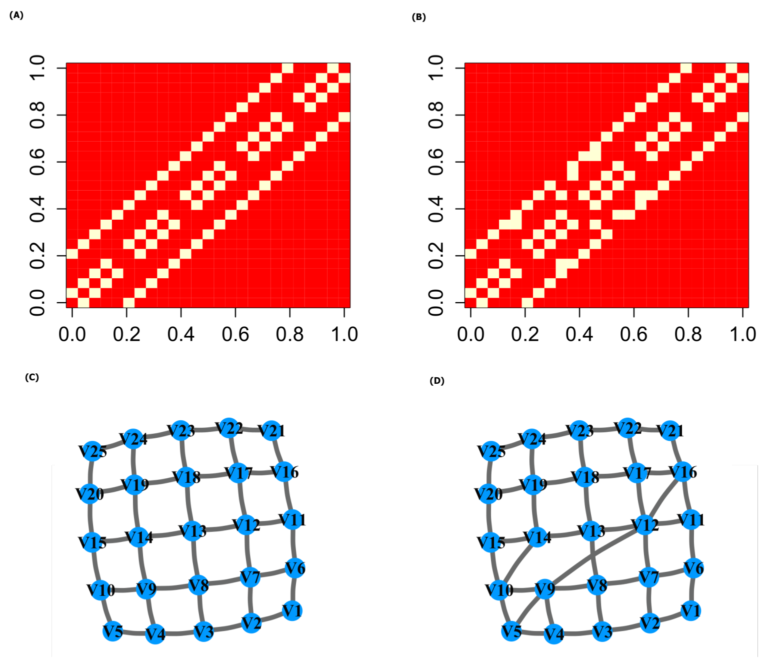


Figure S3: Results of fitting an Ising model to simulated multivariate binary data. The true simulated grid is plotted in (A) and (C). The estimated graph structure via `XMRF(...,method="ISM")` is plotted in (B) and (D).

3.3.2 Estimate LUSC Mutation Networks

In this section, we estimate the relationships among mutated genes in 178 lung squamous cell carcinoma (LUSC) patients from the TCGA project. We obtained the data via `getTCGA` from `TCGA2STAT` package. A total of 13655 genes for LUSC patients were obtained. Genes with a mutation rate of less than 15% in the cohort or with an undefined gene name were filtered out before analysis. This left data with 59 genes and 179 patients. Similar to the work-flow applied on simulated data presented, Ising models were fit across 20 regularization values, and the optimal network was selected from 100 iterations via the StARS approach.

```
> lusc.mut <- getTCGA(disease="LUSC", data.type="Mutation")

> mut.dat <- lusc.mut$dat$gdats
> mut.rate <- apply(mut.dat, 1, sum)/ncol(mut.dat)
> mut.gd <- mut.dat[mut.rate>= 0.15, ]
> mut.gd <- mut.gd[-grep("Unknown", rownames(mut.gd)), ]

> lusc.mut.fit <- XMRF(mut.gd, method="ISM", N=100, nlams=20,
                       stability="STAR", th=0.001, beta=0.1)

> plotGML(lusc.mut.fit, fn="lusc.fit.gml", vars=rownames(mut.gd), weight=TRUE)
```

The estimated mutated gene networks for lung squamous cell carcinoma viewed from Cytoscape is shown in Figure S4.

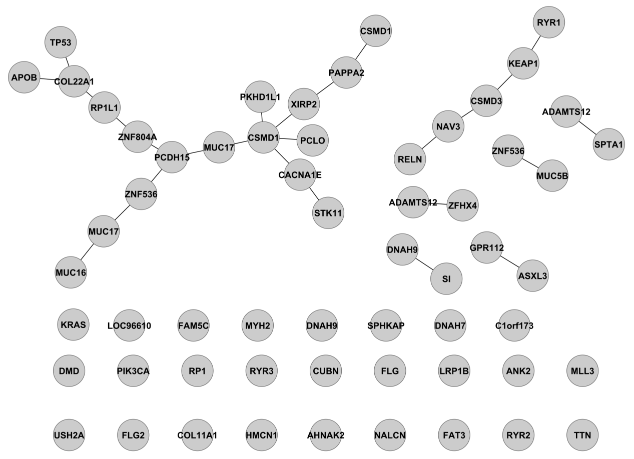


Figure S4: LUSC mutated gene networks estimated by Ising model's `XMRF(...,method="ISM")`

4 Tuning Parameter Selection: Network Sparsity

Our `XMRF` package implements two data-driven methods to determine the sparsity of a fitted network. The first method is the stability selection over many bootstrap samples for a single regularization value (Meinshausen and Bühlmann, 2010). The second method is the StARS selection, which is computed over a range of regularization values to select a network with the smallest regularization value that is simultaneously sparse and reproducible in random samples (Liu *et al.*, 2010). In this section, both of these methods are demonstrated for an example using the local Poisson graphical model (LPGM).

4.1 Stability Selection

Here, we demonstrate the stability selection technique to learn the network sparsity for networks estimated via the `LPGM` method.

We simulate a scale-free network with 30 variables and 200 observations. We determine the network sparsity based on the stability score, which retains network edges that are estimated in more than 95% (`sth=0.95`) of the 50 bootstrap repetitions (`N=50`). The code is given below:

```
> library(XMRF)
> n = 200
> p = 30

# Simulate a scale-free network of 30 notes and 200 samples
> sim <- XMRF.Sim(n=n, p=p, model="LPGM", graph.type="scale-free")
> simDat <- sim$X

# Compute the optimal lambda
```

```
> lmax = lambdaMax(t(simDat))
> lambda = 0.01* sqrt(log(p)/n) * lmax

# Run Local Poisson Graphical Model (LPGM)
> lpgm.fit <- XMRF(simDat, method="LPGM", lambda.path=lambda, sth=0.95, N=50)
```

Results for the code above are shown in Figure S5. It shows that the estimated network structure (Figure S5(B)) is equivalent to the true network structure (Figure S5(A)). Note that stability selection is the default way to determine network sparsity in `XMRF`.

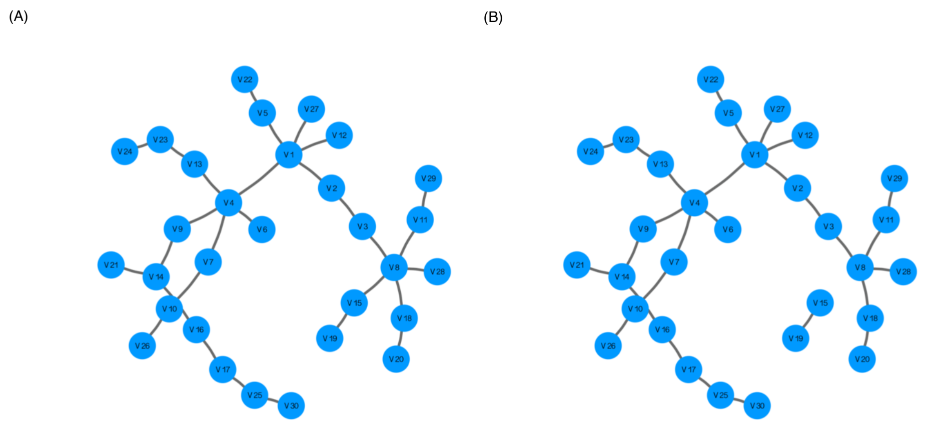


Figure S5: Simulated network from `XMRF.Sim(...,model="LPGM")` (A) and inferred network estimated via `XMRF(...,method="LPGM")` with network sparsity determined via stability selection (B).

4.2 StARS Selection

If users want to fit a network over a series of regularization parameters instead of a single `lambda` as shown in last section, a numerical vector of regularization values should be given for the `lambda.path` parameter of the `XMRF` function.

Another option to study the Markov Networks over the complete regularization path is to let our `XMRF` method decide the path from a null model (empty network) to the full model (saturated network). In this case, the `XMRF(...,method="",...)` function will compute the maximum `lambda` that gives the null model and the minimum `lambda` that gives the full model for each of the parametric familes employed. The maximum `lambda` is computed based on the input data matrix, and is the maximum element from column-wise multiplication of data matrix (data matrix in n x p) normalized by the number of observations. Based on the maximum `lambda` value, the number of `lambda` (`nlams`) and the minimum `lambda` (`lmin`), sequence of appropriate `lambda` values will be computed.

Stability selection via StARS seeks to select the `lambda` value out of the regularization path which yields the most stable network (or, least variable to bootstrap perturbations). Specifically, the variability of each fitted network is measured based on the stability of edges inferred from the bootstrap samples. The network with the smallest penalization and variability below the user specified cutoff (`beta`) is selected as the final optimal network (Liu *et al.*, 2010).

In the following example, we fit the `XMRF(...,method="LPGM")` to learn the same simulated scale-free network of 30 nodes from 200 observations along a path of 20 regularization parameters. Figure S6 shows the printed output of the the following code:

```
> library(XMRF)
> n = 200
> p = 30

# Simulate a scale-free network of 50 notes and 300 samples
> sim <- XMRF.Sim(n=n, p=p, model="LPGM", graph.type="scale-free")
> simDat <- sim$X

# Run LPGM on a whole regularization path
lpgm.fit <- XMRF(simDat, method="LPGM", nlams=20, stability="STAR", th=0.001)
```

5 Visualization and Data Exportation

To enable users to visualize the inferred network in graphical form, `XMRF` includes three plotting functions with slight variations to serve different purposes. First, the default `plot` function of the `GMS` class will draw the optimal inferred network and save it to a PDF file with the following command:

```
> plot(lpgm.fit, fn="lpgm.fit.net.pdf")
```

Second, the `plotNet` function allows users to plot a specific network with specific layout. For example, to plot the simulated network and the inferred network from section 4.1 with the same layout, the following commands can be used (Figure S7):

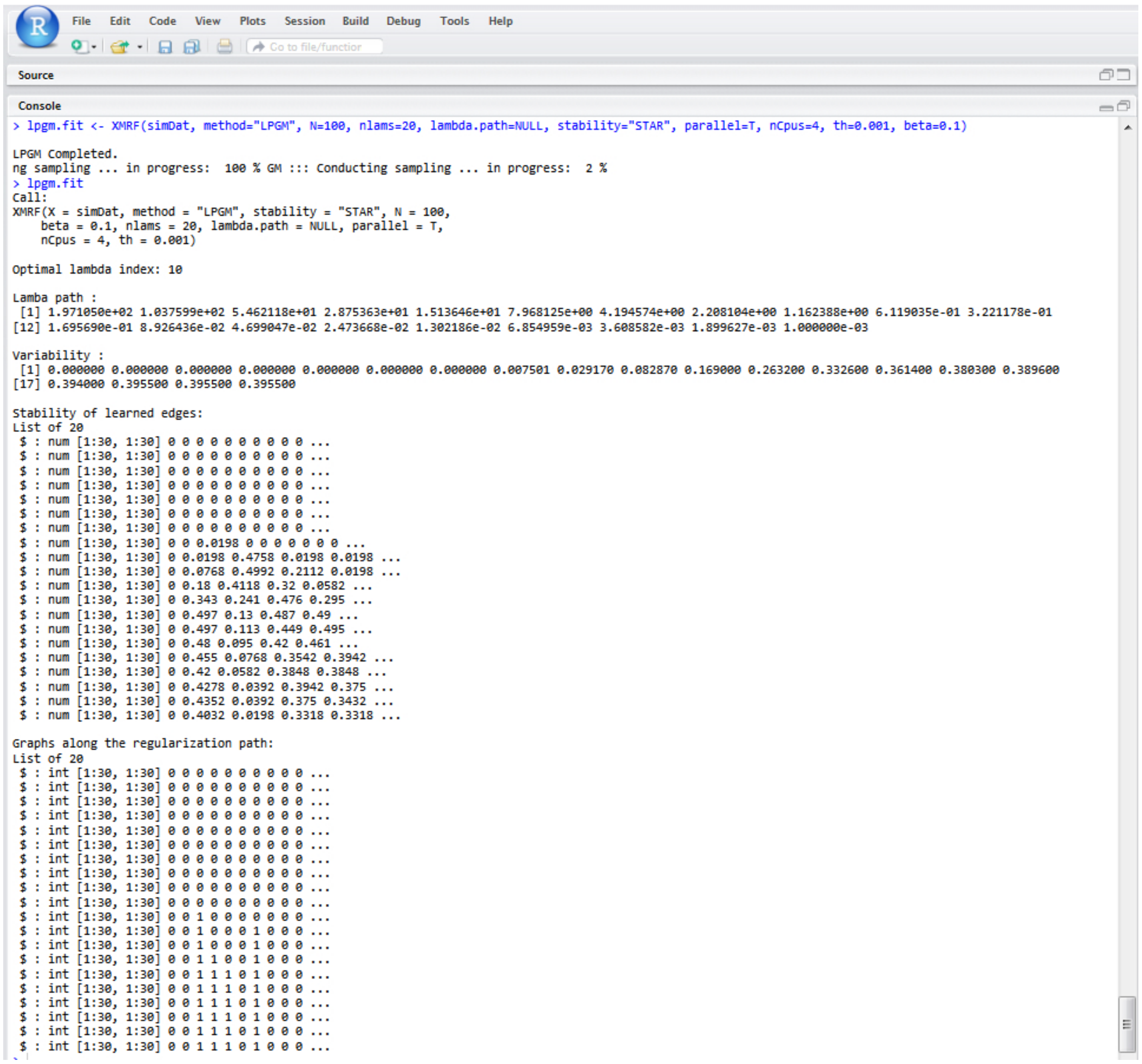
```
> ml = plotNet(sim$B)
> ml = plot(lpgm.fit, mylayout=ml)
```

The third plot function allows users to view the inferred network in other graph visualizing software such as the Cytoscape. The `plotGML` function will write the network in the graph modeling language (GML) format which then can be imported to Cytoscape. For example, with the following command:

```
> plotGML(brca.lpgm, fn="brca.dnet.gml", weight=TRUE, vars=rownames(brca))
```

the estimated BRCA network (stored in `brca.lpgm`) was plotted and viewed from Cytospace (Figure S8). In this example, the width of the edges reflects the stability of the inferred edge.

The inferred network in Figre S8 includes multiple associations reported in published literature, such as the associations of FOXA1, CCND1, and PBX1 with GATA3, link between ERBB2 and CDK12, and others. These results validate the usefulness of the implemented algorithms in the package.



```
> lpgm.fit <- XMRF(simDat, method="LPGM", N=100, nlams=20, lambda.path=NULL, stability="STAR", parallel=T, nCpus=4, th=0.001, beta=0.1)

LPGM Completed.
ng sampling ... in progress:  100 % GM ::: Conducting sampling ... in progress:  2 %
> lpgm.fit
Call:
XMRF(X = simDat, method = "LPGM", stability = "STAR", N = 100,
    beta = 0.1, nlams = 20, lambda.path = NULL, parallel = T,
    nCpus = 4, th = 0.001)

Optimal lambda index: 10

Lamba path :
 [1] 1.971050e+02 1.037599e+02 5.462118e+01 2.875363e+01 1.513646e+01 7.968125e+00 4.194574e+00 2.208104e+00 1.162388e+00 6.119035e-01 3.221178e-01
[12] 1.695690e-01 8.926436e-02 4.699047e-02 2.473668e-02 1.302186e-02 6.854959e-03 3.608582e-03 1.899627e-03 1.000000e-03

Variability :
 [1] 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.007501 0.029170 0.082870 0.169000 0.263200 0.332600 0.361400 0.380300 0.389600
[17] 0.394000 0.395500 0.395500 0.395500

Stability of learned edges:
List of 20
 $ : num [1:30, 1:30] 0 0 0 0 0 0 0 0 0 0 ...
 $ : num [1:30, 1:30] 0 0 0 0 0 0 0 0 0 0 ...
 $ : num [1:30, 1:30] 0 0 0 0 0 0 0 0 0 0 ...
 $ : num [1:30, 1:30] 0 0 0 0 0 0 0 0 0 0 ...
 $ : num [1:30, 1:30] 0 0 0 0 0 0 0 0 0 0 ...
 $ : num [1:30, 1:30] 0 0 0 0 0 0 0 0 0 0 ...
 $ : num [1:30, 1:30] 0 0 0 0 0 0 0 0 0 0 ...
 $ : num [1:30, 1:30] 0 0 0.0198 0 0 0 0 0 0 0 ...
 $ : num [1:30, 1:30] 0 0.0198 0.4758 0.0198 0.0198 ...
 $ : num [1:30, 1:30] 0 0.0768 0.4992 0.2112 0.0198 ...
 $ : num [1:30, 1:30] 0 0.18 0.4118 0.32 0.0582 ...
 $ : num [1:30, 1:30] 0 0.343 0.241 0.476 0.295 ...
 $ : num [1:30, 1:30] 0 0.497 0.13 0.487 0.49 ...
 $ : num [1:30, 1:30] 0 0.497 0.113 0.449 0.495 ...
 $ : num [1:30, 1:30] 0 0.48 0.095 0.42 0.461 ...
 $ : num [1:30, 1:30] 0 0.455 0.0768 0.3542 0.3942 ...
 $ : num [1:30, 1:30] 0 0.42 0.0582 0.3848 0.3848 ...
 $ : num [1:30, 1:30] 0 0.4278 0.0392 0.3942 0.375 ...
 $ : num [1:30, 1:30] 0 0.4352 0.0392 0.375 0.3432 ...
 $ : num [1:30, 1:30] 0 0.4032 0.0198 0.3318 0.3318 ...

Graphs along the regularization path:
List of 20
 $ : int [1:30, 1:30] 0 0 0 0 0 0 0 0 0 0 ...
 $ : int [1:30, 1:30] 0 0 0 0 0 0 0 0 0 0 ...
 $ : int [1:30, 1:30] 0 0 0 0 0 0 0 0 0 0 ...
 $ : int [1:30, 1:30] 0 0 0 0 0 0 0 0 0 0 ...
 $ : int [1:30, 1:30] 0 0 0 0 0 0 0 0 0 0 ...
 $ : int [1:30, 1:30] 0 0 0 0 0 0 0 0 0 0 ...
 $ : int [1:30, 1:30] 0 0 0 0 0 0 0 0 0 0 ...
 $ : int [1:30, 1:30] 0 0 0 0 0 0 0 0 0 0 ...
 $ : int [1:30, 1:30] 0 0 0 0 0 0 0 0 0 0 ...
 $ : int [1:30, 1:30] 0 0 0 0 0 0 0 0 0 0 ...
 $ : int [1:30, 1:30] 0 0 1 0 0 0 0 0 0 0 ...
 $ : int [1:30, 1:30] 0 0 1 0 0 0 1 0 0 0 ...
 $ : int [1:30, 1:30] 0 0 1 0 0 0 1 0 0 0 ...
 $ : int [1:30, 1:30] 0 0 1 1 0 0 1 0 0 0 ...
 $ : int [1:30, 1:30] 0 0 1 1 1 0 1 0 0 0 ...
 $ : int [1:30, 1:30] 0 0 1 1 1 0 1 0 0 0 ...
 $ : int [1:30, 1:30] 0 0 1 1 1 0 1 0 0 0 ...
 $ : int [1:30, 1:30] 0 0 1 1 1 0 1 0 0 0 ...
 $ : int [1:30, 1:30] 0 0 1 1 1 0 1 0 0 0 ...
 $ : int [1:30, 1:30] 0 0 1 1 1 0 1 0 0 0 ...
```

Figure S6: Screen shot of the commands and output of fitting `XMRF(...,method="LPGM")` model over a path of 20 regularization parameters from R studio. The optimal network is the 10th of the 20 networks.

6 From TCGA Data to Gene Networks

To facilitate a smoother pipeline to study gene networks from cancer genomics data, `TCGA2STAT` package is used to provide TCGA data in a format ready for analysis via `XMRF`. Users only need to specify the type of cancer and data platform and the function will query the data from the Broad GDAC Firehose project and process the data into a gene-by-sample matrix format for further analysis.

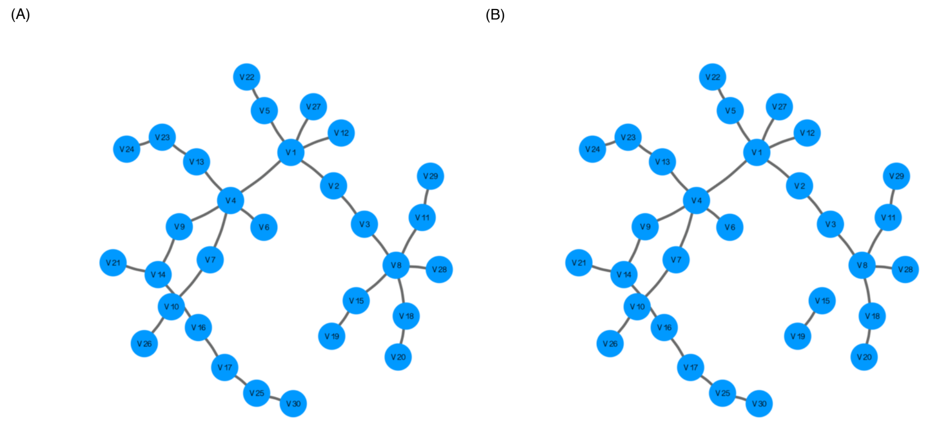


Figure S7: Simulated network from `XMRF.Sim(...,model="LPGM")` (A) and inferred network estimated via `XMRF(...,method="LPGM")` with network sparsity determined via stability selection (B) plotted with the same layout.

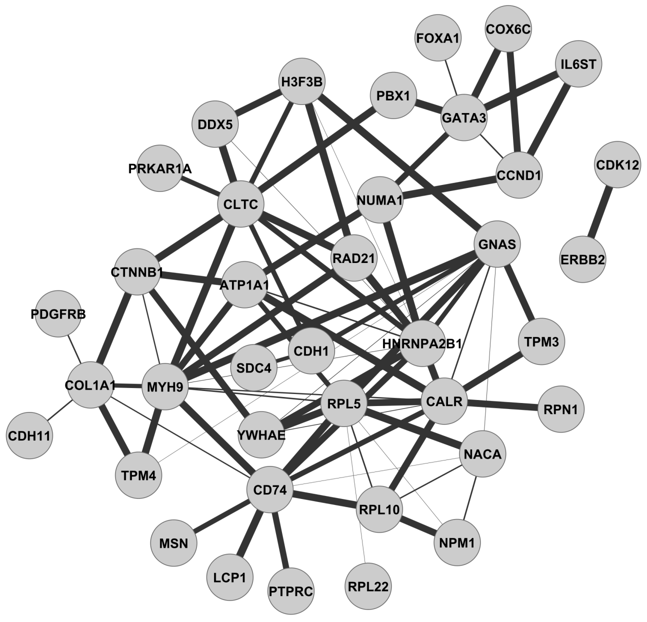


Figure S8: The inferred relationships between cancer census genes from RNASeq data of BRCA patients from TCGA.

6.1 mRNA Microarray Data

This section will give an example of the work-flow of learning gene networks associated with kidney renal clear cell carcinoma (KIRC) from tumor patients:

1. Obtain gene expression data for KIRC, profiled with mRNA microarray platform.

2. Obtain data for only tumor samples.
3. Filter genes so that the top 5% of variable genes remain.
4. Use the XMRF function to learn the network structure. Note that it is always good practice to visualize the data to confirm the distributional family before model fitting. In this example as shown in Figure S9, the data follows a Gaussian distribution and thus fitting a Gaussian Graphical model is appropriate.
5. Write the network in GML format and view the network via Cytoscape (Figure S10).

Code snippets for the above work-flow are provided as follows:

```
> library(TCGA2STAT)

> # Get TCGA data
> # Obtain mRNA array gene expression data for KIRC patients
> kirc.dat <- getTCGA(dataset="KIRC", data.type="mRNA_Array")
> kirc.tum <- SampleSplit(kirc.dat)$primary.tumor

> # Filter genes to remain those of top 5% most variated genes
> var <- apply(kirc.tum, 1, var)
> nac <- apply(kirc.tum, 1, function(x) sum(is.na(x)))
> kirc.tum.gd <- kirc.tum[var >= quantile(var, probs=0.95, na.rm=T) & !is.na(var) & nac==0, ]

> # Take a look at the data to confirm distribution family
> hist(kirc.tum.gd, breaks=20)

> # Fit the data to Gaussian graphical model
> kirc.tum.fit <- XMRF(kirc.tum.gd, method="GGM", N=100, stability="STAR", nlams=10, beta=0.001)

> # Visualized the gene network
> plotGML(kirc.tum.fit, fn="kirc.tum.array.gml", i=2, weight=TRUE, vars=rownames(kirc.tum.gd))
```

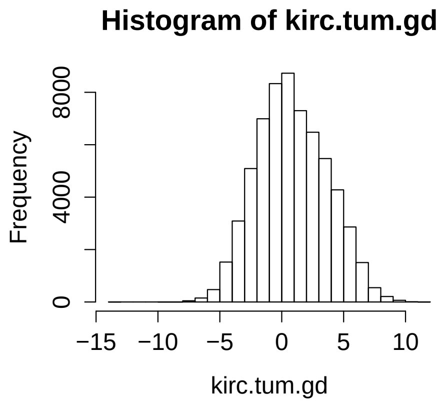


Figure S9: Distribution of mRNA expression profiled with micrarray from KIRC tumor samples.

References

Allen, G. and Liu, Z. (2012). A log-linear graphical model for inferring genetic networks from high-throughput sequencing data. In *Bioinformatics and Biomedicine (BIBM), 2012 IEEE International Conference on*, pages 1–6.

Allen, G. I. and Liu, Z. (2013). A Local Poisson Graphical Model for Inferring Networks From Sequencing Data. *NanoBioscience, IEEE Transactions on*, **12**(3), 189–198.

Csardi, G. and Nepusz, T. (2006). The igraph software package for complex network research. *Inter-Journal*, **Complex Systems**, 1695.

Friedman, J., Hastie, T., and Tibshirani, R. (2010). Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, **33**(1), 1–22.

Knaus, J. (2013). *snowfall: Easier cluster computing (based on snow).* R package version 1.84-4.

Liu, H., Roeder, K., and Wasserman, L. (2010). Stability approach to regularization selection (stars) for high dimensional graphical models. In J. Lafferty, C. Williams, J. Shawe-taylor, R. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 1432–1440.

Meinshausen, N. and Buhlmann, P. (2006). High-dimensional graphs and variable selection with the lasso. *The Annals of Statistics*, **34**(3), 1436–1462.

Meinshausen, N. and Bühlmann, P. (2010). Stability selection. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **72**(4), 417–473.

R Core Team (2013). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Ravikumar, P., Wainwright, M., and Lafferty, J. (2010). High-dimensional ising model selection using l1-regularized logistic regression. *The Annals of Statistics*, **38**(3), 1287–1319.

Yang, E., Ravikumar, P. D., Allen, G. I., and Liu, Z. (2012). Graphical Models via Generalized Linear Models. *NIPS*, pages 1367–1375.

Yang, E., Ravikumar, P., Allen, G. I., and Liu, Z. (2013a). On graphical models via univariate exponential family distributions. *arXiv preprint arXiv:1301.4183*.

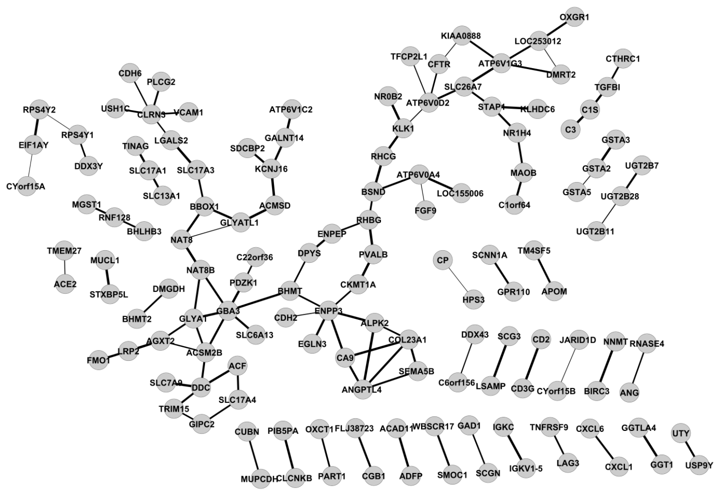


Figure S10: KIRC expressed gene networks estimated by GGM via `XMRF(...,method="GGM")` for mRNA expression data.

Yang, E., Ravikumar, P. D., Allen, G. I., and Liu, Z. (2013b). On Poisson Graphical Models. *NIPS*, pages 1718–1726.