
KINC Documentation

Release v3.4.2

Ben Shealy, Josh Burns, Stephen Ficklin et al

Jul 21, 2020

Contents:

1	About KINC	3
1.1	First, What is a Gene Co-expression Network (GCN)?	3
1.2	How are Traditional GCNs Constructed?	4
1.2.1	Input data	4
1.2.2	Step 1: Pairwise Correlation	4
1.2.3	Step 2: Thresholding	4
1.2.4	Step 3: Module discovery	4
1.3	Problems with GCNs	5
1.3.1	Inappropriate Use of Correlation	5
1.3.2	Bias in Samples	5
1.3.3	High Thresholds	5
1.3.4	Condition-Specific Relationships are Limited	5
1.4	How Does KINC Address These Problems?	6
1.5	Computational Requirements	6
1.6	How was KINC created?	7
2	Functional Overview	9
3	Data Format Overview	13
3.1	Gene Expression Matrix (GEM)	13
3.2	Sample Annotation Matrix (AMX)	14
3.3	Binary Output Files	14
3.3.1	Expression Matrix (EMX)	14
3.3.2	Correlation Matrix (CMX)	14
3.3.3	Cluster Composition Matrix (CCM)	14
3.3.4	Condition Specific Matrix (CSM)	15
3.4	Plain-text Output Files	15
3.4.1	Network File	15
3.4.2	Correlation Matrix	16
4	Prepare Input Data	17
4.1	Create the GEM	17
4.2	Normalizing the GEM	17
4.3	Create the Annotation Matrix	17
5	Installation	19
5.1	Dependencies	19

5.2	Installing KINC on Ubuntu 18.04	20
5.2.1	Install Dependencies	20
5.2.2	Install StatsLib and GCEM	20
5.2.3	Install ACE	21
5.2.4	Install KINC	21
5.2.5	Preparing to Run KINC	22
5.3	Installing on Windows	23
5.4	Installing on an HPC System	23
5.4.1	Palmetto	23
6	Usage	25
6.1	What KINC program to use?	25
6.2	Using the Command-Line	26
6.2.1	Getting Started	26
6.2.2	Executing a Function	27
6.2.3	Using Multiple CPUs	28
6.2.4	GPU Performance Considerations	28
6.2.5	Global Settings	29
6.2.6	Accessing Metadata	30
6.3	Using the Graphical Interface	32
6.3.1	Executing a Function	34
6.3.2	Viewing Help	36
6.3.3	Global Settings	36
6.3.4	Viewing files	36
6.3.5	Accessing Metadata	40
6.4	Using KINC with Docker	43
6.4.1	Interactive Mode	43
6.4.2	Test The Example Data	43
6.4.3	Using the 3D Visualization Tool	44
6.5	Automating KINC with Nextflow	44
7	How to Create a Network	45
7.1	Before Getting Started	45
7.1.1	Traditional or GMM Approach?	45
7.1.2	How Many Samples are Needed?	46
7.1.3	Do Replicates Matter?	46
7.1.4	Which Correlation Method to Choose?	47
7.2	Traditional Approach	47
7.2.1	Step 1: Import the GEM	47
7.2.2	Step 2: Perform Correlation Analysis	47
7.2.3	Step 3: Thresholding	48
7.2.4	Step 4: Extracting the Network File	50
7.3	GMM approach	50
7.3.1	Step 1: Import the GEM	51
7.3.2	Step 2: Perform GMM + Correlation Analysis	51
7.3.3	Step 3: Filter Low-Powered Edges	52
7.3.4	Step 4: Condition-Specific Filtering	52
7.3.5	Step 5: Extract Condition-Specific Subgraphs	53
7.3.6	Step 6: Remove Biased Edges	54
7.3.7	Step 7: Generate Summary Plots	55
7.3.8	Step 8: Threshold the Network by Ranks	56
7.3.9	Step 9: Visualization	57
8	Auxiliary Scripts	61

8.1	kinc.sh	61
8.2	kinc-py.sh	61
8.3	make-input-data.py	61
8.4	validate.py	62
8.5	visualize.py	62
9	Troubleshooting	63
9.1	General Help and Guidance	63
9.2	Thresholding Large GEMs	63
10	How to Cite KINC	65
11	Publications Using KINC	67



The Knowledge Independent Network Construction (KINC) software is a C++ application for constructing [Gene Co-expression Networks \(GCN\)](#) from gene expression data.



1.1 First, What is a Gene Co-expression Network (GCN)?

A Gene Co-expression Network **GCNs** is a **top-down** Systems Biology model of potential gene product interactions. The GCN is modeled as a graph consisting of nodes (i.e. vertices) and connections between them called edges. We refer to these graphs as **networks** in an applied setting such as when exploring gene expression. In a network, when an edge exists between two nodes it carries information. In the case of GCNs, where nodes are genes, the edges indicate correlation of expression. It has been shown that correlation of expression does imply that two genes may be co-functional. This is known as **guilt-by-association**. Therefore, when groups of highly connected (i.e. highly correlated) genes appear in the GCN and tend to be less connected to other genes in the network (i.e. they form a **module** or cluster), **new hypotheses** can be formed suggesting that those highly connected genes, especially genes of unknown function, are co-functional.

1.2 How are Traditional GCNs Constructed?

GCNs, first referred to as relevance networks, have been constructed for approximately two decades. They use the same gene expression data used by Differential Gene Expression (DGE) analysis. However, unlike DGE analysis which only provides a list of candidate differentially expressed genes, GCNs provide potential **context** or “neighborhood” in which those genes may be interacting. However, GCNs typically require more samples than a DGE experiment.

Construction of GCNs traditionally consist of these steps:

- Preparation of input data.
- **Pairwise correlation** of all genes with every other gene.
- **Thresholding** of non-significant correlation values.
- **Module discovery**.

Each of these steps is described in the following sections.

1.2.1 Input data

To construct a GCN, the gene expression data is often organized into a Gene Expression Matrix (GEM). A GEM is a tab delimited file, contain an $n \times m$ data matrix where n is the number of genes and m is the number of measurements (or samples). The GEM is typically normalized and log2 transformed gene expression data from either RNA-Seq or microarray data. The tool [GEMmaker](#) (a sister tool of KINC) can help create GEMs from RNA-Seq data using tools such as [Hisat2](#), [Kallisto](#) or [Salmon](#).

1.2.2 Step 1: Pairwise Correlation

For traditional network construction, each gene in the GEM undergoes correlation analysis with every other gene. This step can use any correlation step. The most popular have been Pearson, Spearman and Kendall Tau. Others have substituted a correlation test for the Information Theory technique of Mutual Information (MI). For any of these, this requires $n(n-1)/2$ correlation calculation, where n is the number of genes. The end-result is an $n \times n$ similarity matrix containing the correlation values of each gene with every other gene. This matrix can become quite large depending on the number of genes.

1.2.3 Step 2: Thresholding

The next step in GCN construction is thresholding. At this step, a method is applied to determine a correlation value below which all pairwise comparisons should be ignored. Those above the threshold are kept. There have been a variety of approaches used for thresholding including ad-hoc methods, Random Matrix Theory (RMT), and soft-thresholding, to name a few. Ad-hoc methods apply a reasonable rule such as keeping the top 1000 co-expressed genes. Simply, RMT involves principles of random matrix theory and applies those to the similarity matrix to explore at which correlation value the matrix begins to show signs of appearing more random. And soft-thresholding was made popular by the [WGCNA](#) tool, published in 2008. With soft-thresholding there is no threshold applied but rather all data can participate, in a weighted manner, to module discovery.

1.2.4 Step 3: Module discovery

After all non-significant pairwise correlation values have been removed, or in the case of soft-thresholding, weighted accordingly, modules can be identified. Modules are groups of genes that are very highly connected with one-another and less connected with the remainder of the network. Modules in networks tend to participate in the same role. For genes, it implies co-functionality. There are many methods for module discovery. These include clustering by nodes

such as with [MCL](#) (Markov Clustering) and WGCNA; and clustering by edges such as with [Linked Communities](#). WGCNA provides module discovery in combination with the soft-thresholding approach.

1.3 Problems with GCNs

1.3.1 Inappropriate Use of Correlation

Despite their increasing use and successes, GCNs constructed in the traditional approach suffer from a variety of problems that if corrected can improve the discoverability that GCNs can afford. First, blanket application of a single correlation method for all gene-expression relationships is inappropriate. Pearson correlation requires that a variety of assumptions are met, including equal variance (homoscedastic), no outliers, and linearity. Spearman is similar except it is less susceptible to outliers and allows for non-linear relationships as long as they are monotonically increasing. Mutual Information approaches have been shown to not perform any better than traditional correlation tests. In any case, when GEMs contain data from multiple experiments with varying conditions, the likelihood that the pairwise comparison of genes will follow the test assumptions decreases. With the decreasing cost of DNA sequencing, and increasing size of experiments, GCNs will be less effective for larger experiment sizes.

1.3.2 Bias in Samples

Second, traditional approaches to GCN construction are biased towards genes whose expression is correlated across almost all the samples. These tend to be basal processes such as transcription, translation, and ribosomal activity, or activities specific to the experimental condition that dominates the GEM. The result, therefore, are GCNs that consist primarily of basal processes and relationships specific to the dominant condition. Genes that are pleiotropic (participate in multiple function) whose expression changes dependent on the condition will most likely be absent from the network. When a large compendium (hundreds to thousands) of samples are used to create a **global** GCN, the result is biased towards these basal activities or over represented conditions in the data set.

1.3.3 High Thresholds

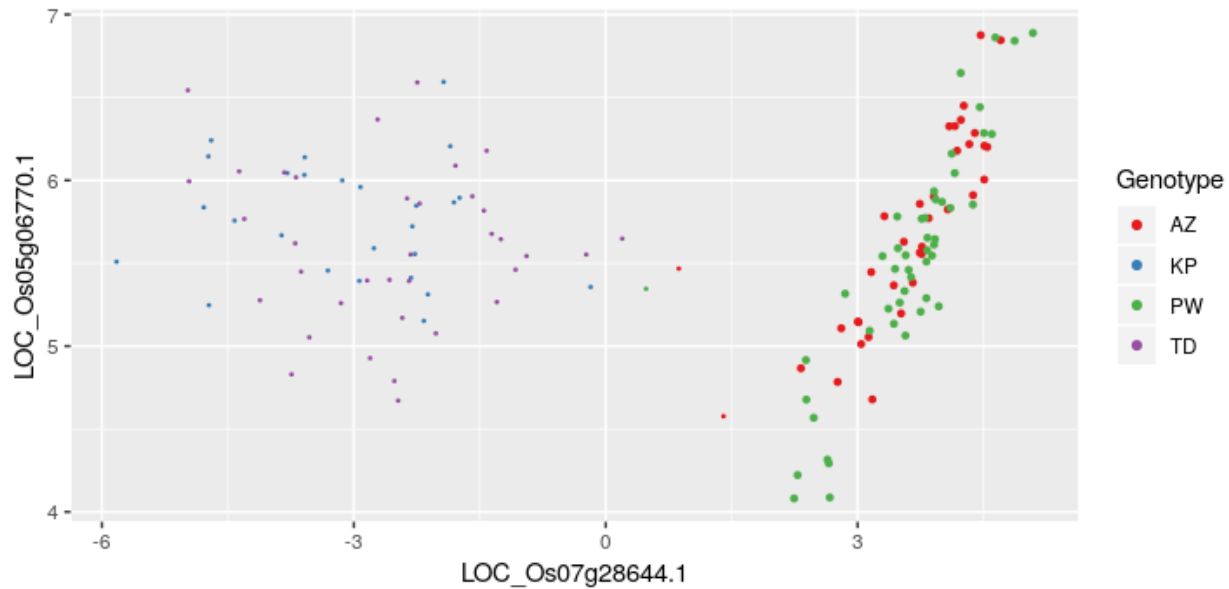
In all traditional network construction approaches, the thresholding step often sets at a very high correlation value. For example, Random Matrix Theory (RMT) uses properties of the co-expression similarity matrix to threshold the network. When applied to similarity matrices generated using Pearson or Spearman correlation, the threshold is often at or above 0.85. In many cases the threshold is above 0.9. A large number of meaningful relationships are therefore excluded. This implies a high level of noise (false positives and false negatives) that limit discovery of gene relationships to only the most highly correlated that transcend a very high level of noise. GCNs whose threshold is cut too low, become unmanageable “hairballs” and when cut very highly, important potential interactions are lost.

1.3.4 Condition-Specific Relationships are Limited

Identification of relationships in the network that are specific to a condition such as developmental stage, tissue, genotype, experimental treatment, environment, etc., are limited due to the reasons mentioned above. Researchers are forced to subdivide their samples into smaller groups that are limited to single conditions, create GCNs for each group and then compare the networks to identify condition-specific relationships or to view common relationships across all. Yet, the noisy nature of GCNs can make network comparisons difficult and limit the discovery of condition-specific subgraphs within the GCN.

1.4 How Does KINC Address These Problems?

KINC, is an evolving tool that attempts to address the above mentioned challenges (and others as they are identified) with the objective of making GCNs more powerful in an age of increasingly large experiments consisting of multiple experimental conditions. One important approach KINC uses to address these challenges, is [Gaussian Mixture Modules](#) (GMMs) to identify groups, or clusters, of similarity expressed genes in the pairwise comparison. Clusters of samples have been observed in 2D plots of pairwise gene expression, and the hypothesis for the presence of these clusters is that they are the result of condition-specific gene expression. Consider the example in the following image:



Here, two genes from *Oryza sativa* (rice) show two “groupings” or clusters of samples. The cluster to the right shows two genotypes (AZ and PW) that are clearly correlated. The cluster to the left is not. GMMs can be used to identify these two groups and then apply a traditional correlation test to each cluster individually.

The use of GMMs addresses some of the challenges previously described. First, the clusters follow a Gaussian distribution, thus, the assumptions of both Pearson and Spearman are met. This reduces the false edges or missing edges that occur when the correlation tests are applied incorrectly. Second, over representation of a condition in the sample set does not bias against under-represented conditions so long as there are enough samples to identify a condition-specific clusters.

Third, condition-specific subgraphs can be identified by using knowledge of the input samples. KINC uses a hypergeometric test for categorical data and regression analysis for quantitative data to compare the experimental conditions or phenotypes for each sample with the samples in a cluster. The p -value and r -squared results from these tests are added to each edge in the network. Edges can then be filtered into smaller **condition-specific subgraphs**.

1.5 Computational Requirements

KINC can be run on a stand-alone Linux desktop or a High Performance Computing (HPC) cluster. Traditional network construction (without GMMs) can easily be performed on a stand-alone machine. However, depending on the number of genes and samples, use of GMMs may require access to Graphical Processing Units (GPUs), and as the size of the GEM grows larger, KINC may require multiple GPUs. A table is provided in the [Usage](#) section to help you decide what type of computational resource you may need.

KINC provides a graphical interface, named *qkinc* that can be used on a stand-alone machine for importing of the GEM file, thresholding, viewing result files, and network file export. The graphical version can be used for similarity

matrix calculation using GMMs only when the sample size is relatively small.

1.6 How was KINC created?

KINC is built with [ACE](#), a framework which provides mechanisms for large-scale heterogeneous computing and data management. As such, KINC can be run in a variety of compute configurations, including single-CPU / single-GPU and multi-CPU / multi-GPU, and KINC uses its own binary file formats to represent the data objects that it consumes and produces. Each of these binary formats can be exported to a plain-text format for use in other applications.

Functional Overview

KINC provides two executables: `kinc` (the command-line version) and `qkinc` (the graphical user interface (GUI) version). Both command-line and GUI versions can perform all of the same functionality, except that the command-line version can use the [Message Passing Interface](#) (MPI) for inter-process communication on an HPC system. The GUI cannot. Additionally, there are several scripts that accompany KINC that use [KINC.R](#) (a supplemental R package for KINC) or custom Python code. As KINC is an evolving tool, these scripts provide functionality that has not yet been incorporated into the KINC binary.

KINC can construct GCNs using a traditional approach (see the [About KINC](#) section) or using the new GMM approach. To this end, a variety of functions for each step in the GCN construction workflow are provided. The names and descriptions of the KINC functions or scripts are provided below in the order that they might be used for network construction.

Note: The command-line and GUI versions of KINC provide the same set of functions. The names of the command-line tool are provided below but the descriptions are the same for both.

Step 1: GEM Import

- `import-emx`: a KINC function that imports a gene expression matrix, or GEM, into a binary format. The binary format is readable by other KINC functions.

Step 2: Similarity Matrix Construction

- `similarity`: a KINC function responsible for creating the similarity matrix and supports both traditional and GMM approaches for GCN construction. It uses the GEM file imported using the `import-emx` function.

Step 3: Filtering

- `corrpwr`: a KINC function that performs [power analysis](#) and removes edges from the network that have insufficient power. This function is only necessary when the number of samples in a cluster are allowed to be small. The minimum size cluster can be set using the `similarity` step. For small clusters, the Type I and Type II error rates are higher and this function ensures that the only edges that remain in the final network are those with a sufficient alpha (default 0.001) and beta (default 0.8) values. Skip this function if the minimum cluster size was set to 30 or greater.

- `cond-test`: a KINC function that performs a hypergeometric test to identify an edge in the network (i.e. cluster identified by GMMs) that is condition-specific. It requires an “Annotation Matrix”: a tab-delimited file where the rows are samples and each column contains some feature about the sample, such as experimental conditions or phenotypic values. Skip this function if GMMs were not used in the `similarity` step. This function is **only applicable to the GMM approach** of network construction and is the primary method for identification of condition specific subgraphs.

Step 4: KINC-based Thresholding

- `rmt`: This function provides the Random Matrix Theory (RMT) approach to thresholding a network. It iterates through decreasing correlation scores, the resulting similarity matrix is examined for properties of random matrices. If the resulting matrix ceases to look non-random a threshold is identified. It is not fully compatible with the GMM approach. **This is the recommended thresholding approach for traditional network construction.**
- `powerlaw`: This thresholds the network at the point that it appears *scale-free*. Most real networks are scale free and a network that follows a power-law distribution is known as a scale-free. This approach iterates through decreasing correlation scores, and checks if the resulting network follows a power-law distribution based on node degree. A threshold can be identified at a given correlation score where the resulting network does not appear scale-free.

Step 5: Network Extraction

- `extract`: Once the network construction steps are completed, this program extracts the network into a tab-delimited text file or GraphML file for use by other tools such as [Cytoscape](#).

Step 6: Post-Network filtering

- `filter-condition-edges.R`: a KINC.R script that removes two types of biased (or false) edges that can appear in the GMM derived networks: edges due to lack of differential cluster expression (DCE) or unbalanced missing data between genes in a comparison. This script is **only applicable to the GMM approach** of network. *Note*: This functionality is in a script until it can be incorporated with the other filter functions of KINC.

Step 7: Post-Network Thresholding

- `rank-condition-threshold.R`: a KINC.R script that organizes the edges from a conditional network, ranks them by *p*-value, similarity score, and `R:sup:2` values (for quantitative data) and keeps the top *n* edges. This is useful for very large condition-specific networks (usually caused by time-series conditions). This script is **only applicable to the GMM approach** of network. *Note*: This functionality is in a script until it can be incorporated with the other threshold functions of KINC.

Step 8: Network Reports & Visualization

- `make-summary-plots.R`: a KINC.R script that generates publication-quality images describing the relationship between the experimental conditions, similarity score, *p*-values and `R:sup:2`. These plots are useful for understanding how condition-specific relationships differ. This script is **only applicable to the GMM approach** of network.
- `view3D-KINC-tidy.py`: a Python Dash application that uses Plotly to create 3D interactive viewer of the network. It allows you to explore the co-expression relationships of each edge. It requires a variety of python libraries to be installed prior to use. This script is **only applicable to the GMM approach** of network.

Other Useful Functions

- `export-emx`: Export a gene expression matrix, or GEM, from the binary format used by KINC into a tab-delimited format compatible with other tools.
- `export-cmx`: Export a correlation matrix (or similarity matrix) from the binary format that KINC uses into a text-based version.

- `import-cmx`: Import a correlation matrix (or similarity matrix) that was previously exported by KINC.

Data Format Overview

KINC generates several custom “data objects” stored as binary files. These files represent the data types that KINC consumes and produces. Additionally, KINC can convert these objects to and from more conventional plain-text formats. Each data object and plain-text format is described here.

3.1 Gene Expression Matrix (GEM)

The GEM is the starting point for all network construction. It is a tab-delimited text file containing an $n \times m$ matrix of gene expression values where rows (or lines in the file) are genes, columns are samples and elements are the expression values for a gene in a given sample.

The GEM may have a header line. But the header line must only contain the sample names. Each subsequent line will always start with the gene name, followed by the expression values. Therefore, the header line, if present, should always have one less element than every other row.

Note: The header line, if present, should always have one less element than every other row. Missing (NaN) values are often represented by a special token such as “NA” or “0.00”.

A very simple example is shown below:

Sample1	Sample2	Sample3	Sample4	
Gene1	0.523	0.991	0.421	NA
Gene2	NA	7.673	3.333	9.103
Gene3	4.444	5.551	NA	0.013

Note: The GEM is imported into KINC and stored in binary format as an Expression Matrix file (described below). KINC can also export the expression matrix back into its text-based GEM format.

3.2 Sample Annotation Matrix (AMX)

The annotation matrix is a tab-delimited or comma-separated file that contains metadata about the RNA-seq samples used to construct the network. The first column should list the sample names (the same as in the GEM) and each additional column should contain information about the sample such as experimental conditions (e.g. Treatment, Tissue, Developmental Stage, Sampling Time, Genotype, etc.) which are usually categorical data. The file can also contain phenotype information that may have been collected from the individuals from which the samples were taken.

Note: The annotation matrix is only needed if condition-specific subgraphs are wanted.

3.3 Binary Output Files

3.3.1 Expression Matrix (EMX)

The Expression Matrix (EMX) file is a data matrix, with genes as rows and samples as columns. It is constructed from the tab-delimited plain-text input Gene Expression Matrix (GEM) file. It contains the matrix data in binary format as well as the gene names and sample names. EMX files have the `.emx` extension.

3.3.2 Correlation Matrix (CMX)

The Correlation Matrix (CMX) file is created during the `similarity` step and is used to represent a similarity matrix, which is a pairwise matrix of similarity (e.g. correlation) scores for each gene pair using data from an expression matrix (EMX). For traditional network construction, the similarity matrix will contain a single similarity score. For the GMM approach, the similarity matrix will contain multiple scores, one for each cluster identified using GMMs. The CMX uses a sparse matrix format—it only stores the gene pairs which have correlation data. Otherwise, the file would grow too large for most file systems. CMX files have the `.cmx` extension.

3.3.3 Cluster Composition Matrix (CCM)

The Cluster Composition Matrix (CCM) file is created during the `similarity` step only when using the GMM approach to network construction. This maintains a matrix identical to the CMX file but instead of similarity scores for each cluster, it stores a “sample composition string”. This string is a series of numbers, strung together indicating which samples belong to the cluster. If a 1 appears in the first position, it indicates the the first sample in the EMX file belongs to the cluster. If a 0 appears it indicates the sample does not. Several other numbers are present to indicate missing or outlier samples. The same is true for the 2nd, 3rd and up to the nth samples. The following indicates the meaning of each number:

- 0: The sample is not present in the cluster.
- 1: The sample is present in the cluster.
- 6: The sample was removed because the expression level in one gene was below the allowed expression threshold.
- 7: The sample was removed as an outlier prior to GMMs.
- 8: The sample was removed as an outlier after the GMM cluster was identified.
- 9: The sample was ignored because one of the genes was missing expression in the sample.

The CCM also uses a sparse matrix format and has a `.ccm` extension.

3.3.4 Condition Specific Matrix (CSM)

The Condition Specific Matrix (CSM) file is created when the `cond-test` thresholding function is used. This thresholding approach can only be used if the GMM approach was used. It requires an “Annotation Matrix” as input, which is a tab-delimited file where the rows are samples and each column contains some feature about the sample, such as experimental conditions or phenotypic values. The file will contain a matrix identical to the CMX matrix but instead of similarity scores, it stores p-values for the association of selected feature from the annotation matrix. The CSM also uses a sparse matrix format and has a `.csm` extension.

3.4 Plain-text Output Files

3.4.1 Network File

There are two types of plain-text output files. First is the **full** format: a tab-delimited file where each line contains information about a single edge in the network. The file contains the following columns:

- **Source:** The first gene in the edge
- **Target:** The second gene in the edge
- **Similarity_Score:** The similarity score value.
- **Interaction:** The interaction type. This always is `co` for co-expression.
- **Cluster_Index:** The unique cluster index (starting from zero)
- **Cluster_Size:** The total number of samples in the cluster.
- **Samples:** The sample composition string.

The following is a sample line from a network file:

Source	Target	Similarity_Score	Interaction	Cluster_Index	Cluster_Size	Samples
Gene1	Gene2	0.979	co	0	30	↪119999191111116111111161116111111111770080000000

Additionally, if the `cond-test` function was performed, a series of additional columns will be present containing the p-values for each test performed. A ‘**Tidy**’ https://en.wikipedia.org/wiki/Tidy_data format for the `cond-test` results can also be exported. **The Tidy format is the recommended format.**

The second format is the **minimal** format, which does not contain the sample string or summary statistics. This format is useful for inspecting large networks quickly. The following is a sample line of a minimal network file:

Source	Target	sc	Cluster	Num_Clusters
Gene1	Gene2	0.979	0	1

The second major network file format is the GraphML format. This is a common XML format used for representing networks. The following is an example snippet of a GraphML file generated by KINC:

```
<?xml version="1.0" encoding="UTF-8"?>
<graphml xmlns="http://graphml.graphdrawing.org/xmlns"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://graphml.graphdrawing.org/xmlns/1.0/graphml.xsd">
  <graph id="G" edgedefault="undirected">
    <node id="Gene1"/>
    <node id="Gene2"/>
    <edge source="Gene1" target="Gene2" samples=
↪"119999191111116111111161116111111111770080000000"/>
  </graph>
</graphml>
```

(continues on next page)

(continued from previous page)

```
</graph>
</graphml>
```

3.4.2 Correlation Matrix

A plain-text correlation matrix is a representation of a sparse matrix where each line is a correlation. It includes the pairwise index, correlation value, sample composition string, and several other summary statistics. The following is a sample line from the correlation matrix file:

```
0          1          0          1          30          5          2          1          3          0.979  ↵
↵1199991911111116111111161116111111111770080000000
```

4.1 Create the GEM

Before using KINC, you must have a valid Gene Expression Matrix (GEM) file. Please see the [Data Format Overview](#) section for a description of the GEM file format. The GEM file can be created however is easiest for you. If you are working with raw RNA-seq FASTQ files or would like to use RNA-seq data from the [NCBI sequence read archive](#) (SRA) then you may consider using [GEMmaker](#). GEMmaker is a sister tool of KINC, and is a Nextflow workflow designed to process large-scale RNA-seq datasets yielding a GEM file. It can create GEMs using tools such as [Hisat2](#), [Kallisto](#) or [Salmon](#).



4.2 Normalizing the GEM

If the GEM contains raw FPKM, RPKM or TPM values directly from the gene quantification tool, you must *log2* transform the data. This can be performed easily using R or Python. Sometimes, the samples must also undergo normalization. Quantile normalization is often performed, but the most appropriate form of normalization is an open research question.

4.3 Create the Annotation Matrix

To generate condition-specific subgraphs you should also prepare a sample annotation matrix. Please see the [Data Format Overview](#) section for a description of the file format. In short, this file should contain all of the experimental information about each sample as well as any phenotypic data about the individuals from which the samples were taken.

This page provides step-by-step instructions to download, compile and install KINC as well as its dependencies on a stand-alone Ubuntu 18.04 system. However, the same set of dependencies are required for installation on any other platform or on a High Performance Computing (HPC) platform.

Even if you install KINC on an HPC system, you may want to install KINC on a local stand-alone machine so that you can view output files using the *qkinc* graphical interface, export network files more easily or perform other tasks.

Additionally, KINC can also be run in a Docker container. Consult the [Usage](#) section for more information.

5.1 Dependencies

KINC requires the following software packages.

- NVIDIA CUDA Toolkit
- OpenCL
- OpenMPI
- QT
- ACE
- The GNU Scientific Library
- OpenBLAS
- LAPACK
- GCEM
- StatsLib
- Boost C++ libraries

Some functionality of KINC (i.e. condition-specific network construction) require two additional set of dependences: KINC.R v1.1 , Python3 and a variety of Python modules for the 3D visualization.

To install KINC.R please follow the installation instructions on the [KINC.R repository](#). KINC.R requires a variety of other R modules.

If you desire to use the Python v3 [Plotly Dash](#) 3D visualization script you must also install the following packages:

- [Numpy](#)
- [Pandas](#)
- [IGraph for Python](#)
- [Plotly](#)
- [Seaborn](#)
- [Dash](#)
- [progress](#)
- [fa2](#)

Install these Python v3 packages using your favorite package manager (i.e. pip, Anaconda, etc.)

5.2 Installing KINC on Ubuntu 18.04

5.2.1 Install Dependencies

Most dependencies are available as packages via Ubuntu and can be installed using the *apt* framework:

```
sudo apt install \  
  qt5-default \  
  libgsl-dev \  
  libopenblas-dev \  
  libopenmpi-dev \  
  ocl-icd-opencl-dev \  
  liblapacke-dev \  
  nvidia-cuda-toolkit \  
  libboost-dev
```

Additionally, since KINC uses the NVIDIA Driver API, you must install either the appropriate NVIDIA drivers for your system or the NVIDIA headless driver if you don't have a GPU:

```
# install NVIDIA driver  
sudo apt install nvidia-driver-435  
  
# install NVIDIA headless driver  
sudo apt install nvidia-headless-435
```

For specific device drivers other than those provided by Ubuntu (i.e. AMD, Intel, NVIDIA, etc), please refer to the manufacturer's website for installation instructions.

5.2.2 Install StatsLib and GCEM

Both StatsLib and GCEM are header-only libraries. To install them, you only need to download the packages and put them where they can be found. The easiest location is in `/usr/local/include` (which requires root access). Below are example installation instructions:

To install StatsLib into `/usr/local/`:

```
git clone -b master --single-branch https://github.com/kthohr/stats ./stats
sudo cp -R ./stats/include/* /usr/local/include
```

To install CGEM into /usr/local/:

```
git clone https://github.com/kthohr/gcem.git ./gcem
sudo cp -R ./gcem/include/* /usr/local/include
```

5.2.3 Install ACE

KINC v3.4 requires ACE v3.2. ACE requires some of the same dependencies as KINC (such as QT, CUDA, OpenMPI, OpenCL, etc). Therefore, if all dependencies above are installed, ACE should compile. To start, set the following environment variable:

```
export ACE_VERSION=v3.2.0
```

Next, clone the ACE repository:

```
git clone https://github.com/SystemsGenetics/ACE.git
cd ACE/build
git checkout $ACE_VERSION
```

Default installation location

Next compile:

```
qmake ../src/ACE.pro
make qmake_all
make
make qmake_all
make install
```

Alternative installation location

By default, ACE will try to install into /usr/local. To install ACE to a different directory (e.g. /local/software), set the INSTALL_PREFIX environment variable accordingly:

```
export INSTALL_PREFIX="/local/software"
```

Now, within the ACE/build directory run the following to build the ACE libraries:

```
qmake ../src/ACE.pro PREFIX=$INSTALL_PREFIX/ACE-$ACE_VERSION
make qmake_all
make
make qmake_all
make install
```

This will install ACE into the directory specified by INSTALL_PREFIX in a directory named with the ACE version.

5.2.4 Install KINC

Select a suitable [version of KINC](#) and set the environment variable:

```
export ACE_VERSION=v3.2.0
export KINC_VERSION=v3.4.2
```

Next, clone the KINC repository:

```
git clone https://github.com/SystemsGenetics/KINC.git
cd KINC
git checkout $KINC_VERSION
```

Default installation location

Next compile:

```
make
make install
```

Alternative installation location

By default, KINC will try to install itself into `/usr/local`. To install KINC to a different directory (e.g. `/local/software`), set the `INSTALL_PREFIX` environment variable accordingly:

```
export INSTALL_PREFIX="/local/software"
```

Now build and install KINC:

```
make
make install
```

If ACE is not in `/usr/local`

If ACE was not installed into an alternative location other than the default `/usr/local` then should set several environment variables help the compiler find ACE libraries and headers:

```
export PATH="$INSTALL_PREFIX/ACE-$ACE_VERSION/bin:$PATH"
export LD_LIBRARY_PATH="$INSTALL_PREFIX/ACE-$ACE_VERSION/lib:$LD_LIBRARY_PATH"
export LIBRARY_PATH="$INSTALL_PREFIX/ACE-$ACE_VERSION/lib:$LIBRARY_PATH"
export CPATH="$INSTALL_PREFIX/ACE-$ACE_VERSION/include:$CPATH"
export C_INCLUDE_PATH="$INSTALL_PREFIX/ACE-$ACE_VERSION/include:$C_INCLUDE_PATH"
export CPLUS_INCLUDE_PATH="$INSTALL_PREFIX/ACE-$ACE_VERSION/include:$CPLUS_INCLUDE_
↪PATH"
export OBJC_INCLUDE_PATH="$INSTALL_PREFIX/ACE-$ACE_VERSION/include:$OBJC_INCLUDE_PATH"
```

5.2.5 Preparing to Run KINC

If KINC was installed in the default location you can skip the *Usage* page for further instructions, otherwise, if you installed KINC in an alternative location, you must update the `LD_LIBRARY_PATH` in your `~/.bashrc` file. Use the following command to get the exact text you need to add.

```
echo "export LD_LIBRARY_PATH=\"$INSTALL_PREFIX/ACE-$ACE_VERSION/lib:$INSTALL_PREFIX/
↪KINC-$KINC_VERSION/lib:$LD_LIBRARY_PATH\""
echo "export PATH=\"$INSTALL_PREFIX/ACE-$ACE_VERSION/bin:$INSTALL_PREFIX/KINC-$KINC_
↪VERSION/bin:$PATH\""
```

(continues on next page)

(continued from previous page)

Append the resulting text to your `~/.bashrc` file. You should now be able to run KINC

5.3 Installing on Windows

Windows is currently not supported because there is no OpenMPI library for the Windows platform. Future support for Windows will be added when MPI becomes an optional dependency.

5.4 Installing on an HPC System

Usage of KINC on high-performance computing (HPC) systems will require assistance of the cluster's systems admin to ensure all dependencies are installed and available. Software management on clusters is specific to each cluster, although there are often commonalities. Regardless, it is not possible to provide comprehensive instructions that would apply to every cluster.

5.4.1 Palmetto

The following instructions are specific to the Palmetto cluster at Clemson University, however they can be adapted with some effort to other HPC clusters.

If you have previously used any version of KINC or ACE, be sure to remove the modules from your libraries. Furthermore, check to make sure that your `.bashrc` is clear of any designated paths for ACE or KINC.

Obtain an interactive node with at least 8 cores. Run the command:

```
qsub -I -l select=1:ncpus=8:ngpus=2:gpu_model=p100
```

Once you have obtained an interactive node, run the following commands from your home directory:

```
git clone https://github.com/bentsherman/pbs-toolkit.git
./pbs-toolkit/modules/install-ace.sh v3.2.0
./pbs-toolkit/modules/install-statslib.sh
./pbs-toolkit/modules/install-kinc.sh v3.4.2 v3.2.0
```

These scripts will install ACE and KINC into your home directory, establishing them as modules that can be run from anywhere. It will also update your environment so that the modules can be called when necessary. It uses a module called `use.own`, which when added will make KINC and ACE available to be used interactively. You should now be able to load KINC as a module:

```
module add use.own
module add KINC/v3.4.2
```


6.1 What KINC program to use?

KINC is meant to be used either on a stand-alone workstation or on a heterogenous cluster of computers. The graphical program, `qkinc`, the command-line tool `kinc` and scripts are meant to be used in different circumstances. The following table indicates when one should be used over the other.

Task	Program or Script	Resource
Viewing binary files created by KINC (emx, cmx, ccm or csm files)	<code>qkinc</code>	Stand-alone machine
Importing GEM	<code>qkinc</code> or <code>kinc</code>	Stand-alone or HPC
Traditional network construction	<code>qkinc</code> or <code>kinc</code>	Stand-alone or HPC
GMM-based network construction for a small GEM (<60 samples, <20K genes)	<code>qkinc</code> or <code>kinc</code>	Stand-alone or HPC. GPU recommended
GMM-based network construction for a large GEM	<code>kinc</code>	HPC only. GPU required.
RMT, Power-law or condition p-value thresholding	<code>qkinc</code> or <code>kinc</code>	Stand-alone or HPC
Network extraction	<code>qkinc</code> or <code>kinc</code>	Stand-alone or HPC
Filter biased condition-specific networks	<code>kinc-filter-bias</code> R	Stand-alone or HPC
Rank-based thresholding of condition-specific networks	<code>kinc-filter-rank</code> R	Stand-alone or HPC
Generate summary plots	<code>kinc-make-plots</code> R	Stand-alone or HPC
3D Visualization	<code>kinc-3d-viewer</code> py	Stand-alone

6.2 Using the Command-Line

6.2.1 Getting Started

Using the `kinc` program will invoke the command-line interface. Running `kinc` without any options will automatically provide default usage instructions for retrieving help:

```
Help: kinc help <command>

Get help about the specific command <command>.

Valid commands:

    run: Run an analytic in normal mode. MPI is automatically detected. If no
        MPI is detected then the analytic will run in single mode.

    chunkrun: Run an analytic in chunk run mode. This mode will execute the
        analytic on a subset of the data and save its results in a temporary
        file.

    merge: Merge all temporary result files from chunkruns into the finished
        output data objects of the analytic run.

    dump: Dump the system or user metadata of a data object to standard output
        as JSON formatted text.

    inject: Inject new user metadata into a data object from a JSON formatted text
        file, overwriting any user metadata the data object currently
        contains.

    settings: Access the global persistent settings for this program to either view
        those settings or change them.
```

You can retrieve a list of all of the functions that KINC provides by executing

```
kinc help run
```

The following will be shown for KINC v3.4.2:

```
Command: kinc run <analytic> <options...>
Runs the given analytic with the given options.

Help: kinc help run <analytic>
Get help about running a specific analytic <analytic>.

Valid analytics:
import-emx: Import Expression Matrix
export-emx: Export Expression Matrix
import-cmx: Import Correlation Matrix
export-cmx: Export Correlation Matrix
export-cpm: Export Parameter Matrix
similarity: Similarity
corrpower: Filter: Correlation Power
cond-test: Threshold: Condition-Specific
powerlaw: Threshold: Power-law
rmt: Threshold: RMT
extract: Extract Network
```


The functions (or analytics) that KINC provides are listed in the `Valid analytics` section. Thus, to import an expression matrix, the function is named `import-emx`.

The instructions also indicate how to get help for each function by calling `kinc help run <analytic>` where `<analytic>` is the name of the function. For example, to retrieve help for the `import-emx` function:

```
kinc help run import-emx
```

which returns:

```
Command: kinc run|chunkrun|merge import-emx <options...>
Run the given analytic in normal, chunk, or merge mode. For chunk and merge
modes all separate executions MUST have the same options provided to the
analytic.

OPTIONS

--input <value>
Value Type: Input File
Input text file containing space/tab delimited gene expression data.

--output <value>
Value Type: Output Data Object
A data file created by KINC containing the gene expression matrix created by the
Import Expression Matrix analytic.

--nan <value>
Value Type: String
Default Value: NA
Expected token for expressions that have no value.

--samples <value>
Value Type: Integer
Minimum Value: 0
Maximum Value: 2147483647
Default Value: 0
Number of samples. 0 indicates the text file contains a header of sample names
to be read to determine size.
```

The output above shows the command-line arguments, the type of value that is accepted and any default values if you do not specify the argument.

Note: Help instructions are accessible on the command-line for every function of KINC.

6.2.2 Executing a Function

Any function (i.e. analytic) in KINC can be executed in the following way:

```
kinc run <function> [<arguments>]
```

Where `<function>` is the name of the function and `[<arguments>]` is a set of arguments as described in the help documentation of the function. Using the help instructions as shown in the previous section, we can import a GEM, named say `rice_heat_drought.GEM.txt` that has a header and missing values represented as NA in the following way:

```
kinc run import-emx --input ./rice_heat_drought.GEM.txt --output ./rice_heat_drought.  
↳GEM.emx --nan "NA"
```

6.2.3 Using Multiple CPUs

With MPI

KINC can use the Message Passing Interface (MPI) to take advantage of multiple CPUs. This includes CPUs on a stand-alone workstation or across an HPC system. To use MPI on a stand-alone workstation you must launch KINC using the `mpiexec` program and specify the number of processes to use with the `-np` argument. For example to launch a function using 4 CPUs:

```
mpiexec -np 4 kinc run <function> [<arguments>]
```

To use MPI on an HPC system, please consult with the HPC admins or the system's usage documentation to properly use MPI as each system may have a different setup and different instructions.

With Chunking

KINC executes a function by dividing the entire task into sub units that can be processed independently of one another. When using MPI, KINC launches a “master” instance and several “worker” instances. The master communicates with the workers and provides them with work units. However, when MPI is not available, it is possible to manually launch multiple instances of `kinc` and instruct each one to work on a different set of work units. To use chunking you must use the command `chunkrun` instead of `run` and provide two additional arguments: `index` and `size`:

```
kinc chunkrun <index> <size> <function> [<arguments>]
```

Here the `<size>` argument is the total number of chunks to use. This should be set to the number of `kinc` processes you wish to run. Then, launch each `kinc` instance with an `index` value starting at 0. Therefore, to split the jobs into four chunks you would run KINC four times each with a different `index`: 0, 1, 2 and 3. and the `size` for each run is 4. Each instance of KINC will automatically know which set of work units to process.

Once all of the KINC instances have completed their chunks, the results must be merged together into a single file. This is accomplished using the `merge` command. You must provide the exact same arguments to the `merge` command as was provided to the `chunkrun` command, with the exception of the `<index>` argument:

```
kinc merge <size> <function> [<arguments>]
```

Note: When using the `chunk` command you can launch as many KINC processes as your computing resources will allow. Just be sure to set the `<size>` argument to match. However, when running the `merge` command you will only launch one instance to merge everything into a single file.

Note: Use of MPI with KINC is much more efficient than the chunking approach. This is because KINC can provide more work units to faster nodes. So, it is best to use MPI when the facility is available.

6.2.4 GPU Performance Considerations

KINC can run with a variety of hardware configurations. It can use multiple CPU cores and multiple GPUs. It can be run on a stand-alone workstation or for larger datasets it can use multiple nodes of a High-Performance Compute

(HPC) cluster. The `qkinc` program can only use one CPU or GPU, but the command-line `kinc` can use multiple. If you have a GPU available on your system you can control performance of GPU-enabled functions (e.g. the similarity function) by tweaking the following parameters for GPU performance:

Note: Reasonable defaults have been set within KINC for the parameters below. Only change these if you fully understand them.

- **CUDA/OpenCL Thread Size:** Determines the number of worker threads per GPU. Increasing this value can increase performance by utilizing the GPU more fully, but setting this value too high can also decrease performance due to the overhead of switching between many threads. A safe value for this parameter is 2 threads, however on newer hardware it may be possible to use more threads and achieve better performance. This parameter is set using the `threads` option in the KINC settings.
- **Global Work Size:** Determines the number of work items that a worker thread processes in parallel on the GPU. It should be large enough to fully utilize the GPU, but setting it too large can also decrease performance due to global memory congestion and work imbalance on the GPU. In practice, the default value of 4096 seems to work the best. This parameter is set using the `--gisize` option in the `similarity` analytic.
- **Local Work Size:** Determines the OpenCL local work size (CUDA block size) of each GPU kernel. In general, the optimal value for this parameter depends heavily on the particular GPU kernel, but since all of the GPU kernels in KINC are memory-intensive, the local work size should be small to prevent global memory congestion. In practice, a value of 16 or 32 (the default) works the best. This parameter is set using the `--lsize` option in the `similarity` analytic.
- **Block Size:** Determines the number of work items per MPI work block. It is effectively the maximum number of work items that a worker thread can process in parallel. In practice, the work block size does not affect performance so long as it is greater than or equal to the global work size, so the default value of 32,768 should work well. This parameter is set using the `--bsize` option in the `similarity` analytic.

6.2.5 Global Settings

KINC maintains a set of global settings. These are parameters that control the behavior of KINC and are persistent between KINC executions. If a setting change is made by one instance of KINC, it is set for all instances. You can see the list of settings provided by KINC by executing the following command:

```
kinc settings
```

The above command results in the following:

```
SETTINGS
    CUDA Device: 0
    OpenCL Device: 0:0
CUDA/OpenCL Thread Size: 4
    MPI Buffer Size: 4
Chunk Working Directory: .
    Chunk Prefix: chunk
    Chunk Extension: abd
    Logging: off
```

The settings and their meaning are described in the following table:

Setting	Description
CUDA Device	The index of the default GPU device for use with the CUDA drivers. This defaults to index 0 on a machine with a GPU
OpenCL Device	The index of the default GPU device for use with the OpenCL drivers. This defaults to index 0:0 on a machine with a GPU
CUDA/OpenCL Thread Size	The number of threads to use for the GPU.
MPI Buffer Size	The size of the MPI buffer when the master and worker nodes communicate
Chunk Working Directory	The directory where the chunk results files will go
Chunk Prefix	The prefix that will be used for all of the chunk files.
Chunk Extension	The extension that will be used for all of the chunk files.
Logging	For debugging purposes, KINC will provide very deep logging. Users need not ever enable logging as this is meant for KINC developers.

To change a setting, use the following command-line:

```
kinc settings set <parameter> <value>
```

For example, to disable the CUDA Device:

```
kinc settings set cuda none
```

Note: Most users will never need to adjust these persistent settings.

6.2.6 Accessing Metadata

KINC strives to ensure reproducibility of results by maintaining system and user metadata within each file. You can access metadata via the command-line for viewing. System meta data maintains a complete provenance for how the file was created. System metadata is immutable. User metadata consists of information about the run of the analytic.

Retrieving System Metadata

To view the system meta data for any KINC file use the following command:

```
kinc dump <file> system
```

Where <file> is the path to a KINC generated file. Metadata will be output in JSON format similar to the following example from an expression matrix (.emx extension) file:

```
{
  "command": {
    "analytic": "Import Expression Matrix",
    "options": {
      "input": "../01-input_data/rice_heat_drought/rice_heat_drought.GEM.
↪FPKM.filtered.txt",
      "nan": "NA",
      "output": "rice_heat_drought.GEM.FPKM.filtered.emx",
      "samples": "0"
    }
  }
}
```

(continues on next page)

(continued from previous page)

```

    },
    "input": {
    },
    "uuid": "{ae169a67-363d-4a8c-8a04-de0fd8d974f8}",
    "version": {
      "ace": {
        "major": 3,
        "minor": 2,
        "revision": 0
      },
      "kinc": {
        "major": 3,
        "minor": 4,
        "revision": 0
      }
    }
  }
}

```

Notice that the metadata provides the exact command-line and arguments that were used to produce the file, as well as a unique UUID for the file and the versions of the ACE and KINC that were used to produce the file.

As KINC files are used in other functions, the system metadata is preserved. Therefore the complete provenance from beginning to end for creation of the file is maintained. Consider the following example of system metadata from a correlation matrix (.cmx extension) file. Notice it has the exact command-line arguments for the `similarity` function, but also includes the system metadata for all of the input files that it used, including the expression matrix.

```

{
  "command": {
    "analytic": "Similarity",
    "options": {
      "bsize": "0",
      "ccm": "rice_heat_drought.GEM.FPKM.filtered.ccm",
      "clusmethod": "gmm",
      "cmx": "rice_heat_drought.GEM.FPKM.filtered.cmx",
      "corrmethod": "spearman",
      "crit": "ICL",
      "gsize": "4096",
      "input": "rice_heat_drought.GEM.FPKM.filtered.emx",
      "lsize": "32",
      "maxclus": "5",
      "maxcorr": "1",
      "minclus": "1",
      "mincorr": "0.5",
      "minexpr": "-inf",
      "minsamp": "25",
      "postout": "TRUE",
      "preout": "TRUE"
    }
  },
  "input": {
    "rice_heat_drought.GEM.FPKM.filtered.emx": {
      "system": {
        "command": {
          "analytic": "Import Expression Matrix",
          "options": {
            "input": "../01-input_data/rice_heat_drought/rice_heat_
↪drought.GEM.FPKM.filtered.txt",

```

(continues on next page)

(continued from previous page)

```

        "nan": "NA",
        "output": "rice_heat_drought.GEM.FPKM.filtered.emx",
        "samples": "0"
    },
    "input": {
    },
    "uuid": "{ae169a67-363d-4a8c-8a04-de0fd8d974f8}",
    "version": {
        "ace": {
            "major": 0,
            "minor": 0,
            "revision": 999
        },
        "kinc": {
            "major": 3,
            "minor": 3,
            "revision": 0
        }
    }
},
<trimmed here for brevity>

```

Retrieving User Metadata

User metadata can be retrieved using a similar command:

```
kinc dump <file> user
```

6.3 Using the Graphical Interface

KINC provides a graphical user interface (GUI) for viewing binary output files and for executing less computationally intensive jobs. The graphical interface is meant to run only on a stand-alone workstation as it cannot launch multiple worker instances as the command-line version can do. This section provides a brief overview of the GUI. To launch `qkinc` simply call it on the command-line:

```
qkinc
```

When the GUI first appears, it is a simple dialog box with a menu:

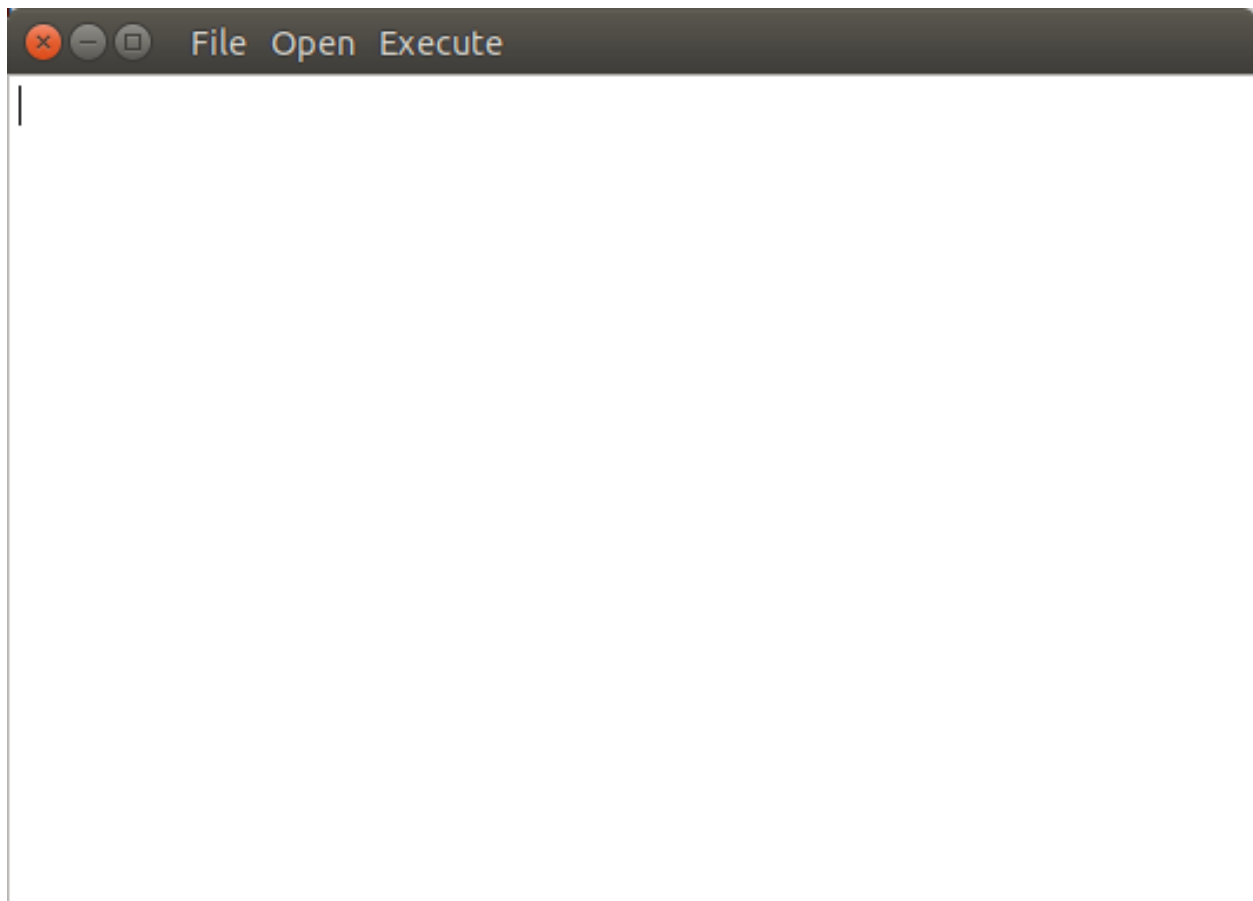
The following items are available in the main menu:

File: for general settings and information.

- *Settings*: Used to adjust KINC's global persistent settings.
- *About*: Provides information about KINC.
- *Exit*: Closes the program.

Open: for opening KINC binary files

- *Expression Matrix*: Opens an expression matrix for viewing.
- *Cluster Matrix*: Opens GMM cluster details for each edge in the network.



- *Correlation Matrix*: Opens the similarity (or correlation) matrix.
- *Condition-Specific Clusters Matrix*: Opens the the matrix containing the results from the Cluster-Specific thresholding.

Execute: for running the functions of KINC

- *Import Expression Matrix*: Imports a GEM. Corresponds to the `import-emx` function.
- *Export Expression Matrix*: Exports a GEM. corresponds to the `export-emx` function.
- *Import Correlation Matrix*: Imports a correlation matrix. Correponds to the `import-cmx` function
- *Export Correlation Matrix*: Exports a correlation matrix. Correponds to the `export-cmx` function.
- *Similarity*: Performs pairwise correlation analysis for both traditional and GMM approaches. Corresponds to the `similarity` function.
- *Filter: Correlation Power*: Performs power analysis to remove edges with low power. Corresponds to the `corrpower` function.
- *Threshold: Condition-Specific*: Performs condition-specific thresholding. Corresponds to the `cond-test` function.
- *Threshold: Power-law*: Performs thresholding using the power-law to ensure a scale-free network. Corresponds to the `powerlaw` function.
- *Threshold: RMT*: Performs thresholding using Random Matrix Theory. Corresponds to the `rmt` function.
- *Extract Network*: Extracts the final method by applying the threshold. Correponds to the `extract` function.

6.3.1 Executing a Function

To execute a function, simply select it from the *Execute* menu. A dialog box will appear providing a form to enter the argumetns for the function. The form for importing a GEM is shown in the following screenshot:

The screenshot shows a dialog box titled "Execute Import Expression Matrix". It contains the following fields and controls:

- Input:** A text field containing "M.FPKM.filtered.txt" and a "Browse" button.
- Output Expression Matrix:** A text field containing "M.FPKM.filtered.emx" and a "Browse" button.
- NAN Token:** A text field containing "NA".
- Sample Size:** A text field containing "0" with a small up/down arrow control to its right.
- Buttons:** "Execute" and "Cancel" buttons at the bottom.

A view of the `similarity` function is shown in the following screenshot:

Execute Similarity

Expression Matrix:

1.FPKM.filtered.emx

Browse

Output Cluster Matrix:

1.FPKM.filtered.ccm

Browse

Output Correlation Matrix:

1.FPKM.filtered.cmx

Browse

Clustering Method:

gmm

Correlation Method:

spearman

Minimum Expression:

-inf

Minimum Sample Size:

30

Minimum Clusters:

1

Maximum Clusters:

5

Criterion:

ICL

Remove pre-outliers:

☒

Remove post-outliers:

☒

Minimum Correlation:

0.5

Maximum Correlation:

1

Work Block Size:

0

Global Work Size:

4096

Local Work Size:

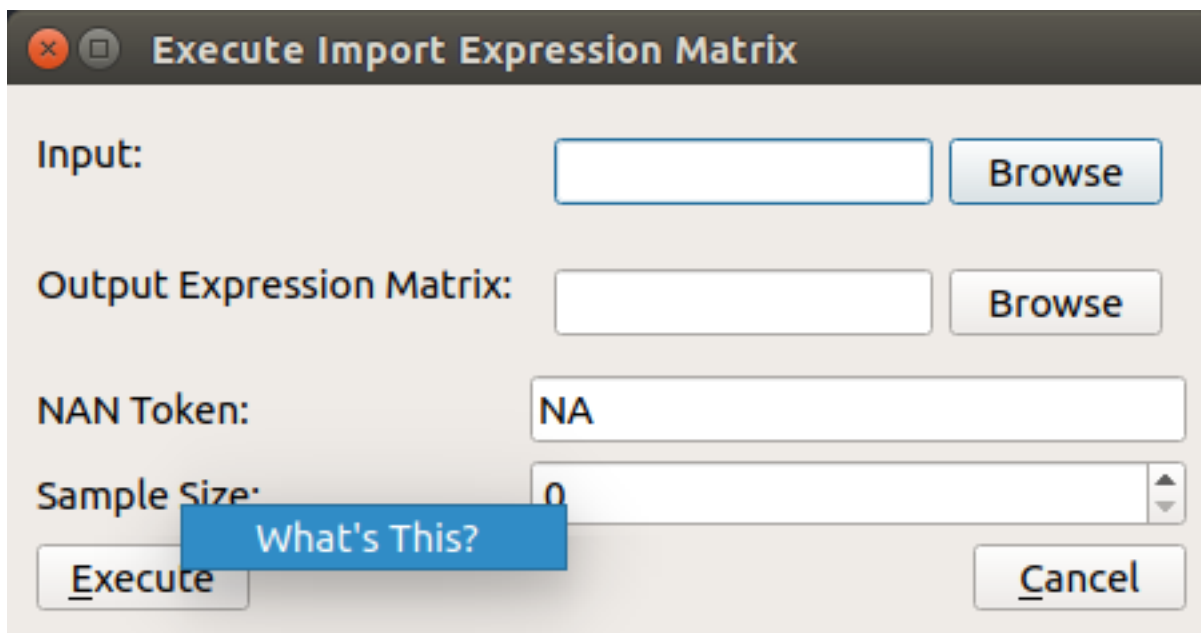
32

Execute

Cancel

6.3.2 Viewing Help

On each form, as shown in the previous two screenshots, more information about each parameter can be obtained by left clicking on the argument. A *Whats this?* toolkit will appear. Click the tooltip to see the help for that parameter.



6.3.3 Global Settings

As previously described in the *Global Settings* section for the *Command-line Usage*, KINC provides a set of persistent global settings that remain set even when the KINC GUI is closed. Settings changes made on the command-line or via the GUI are persistently the same for both the command-line and GUI versions. You can view and change the global settings via the **File > Settings** menu. A view of the settings form is shown below:

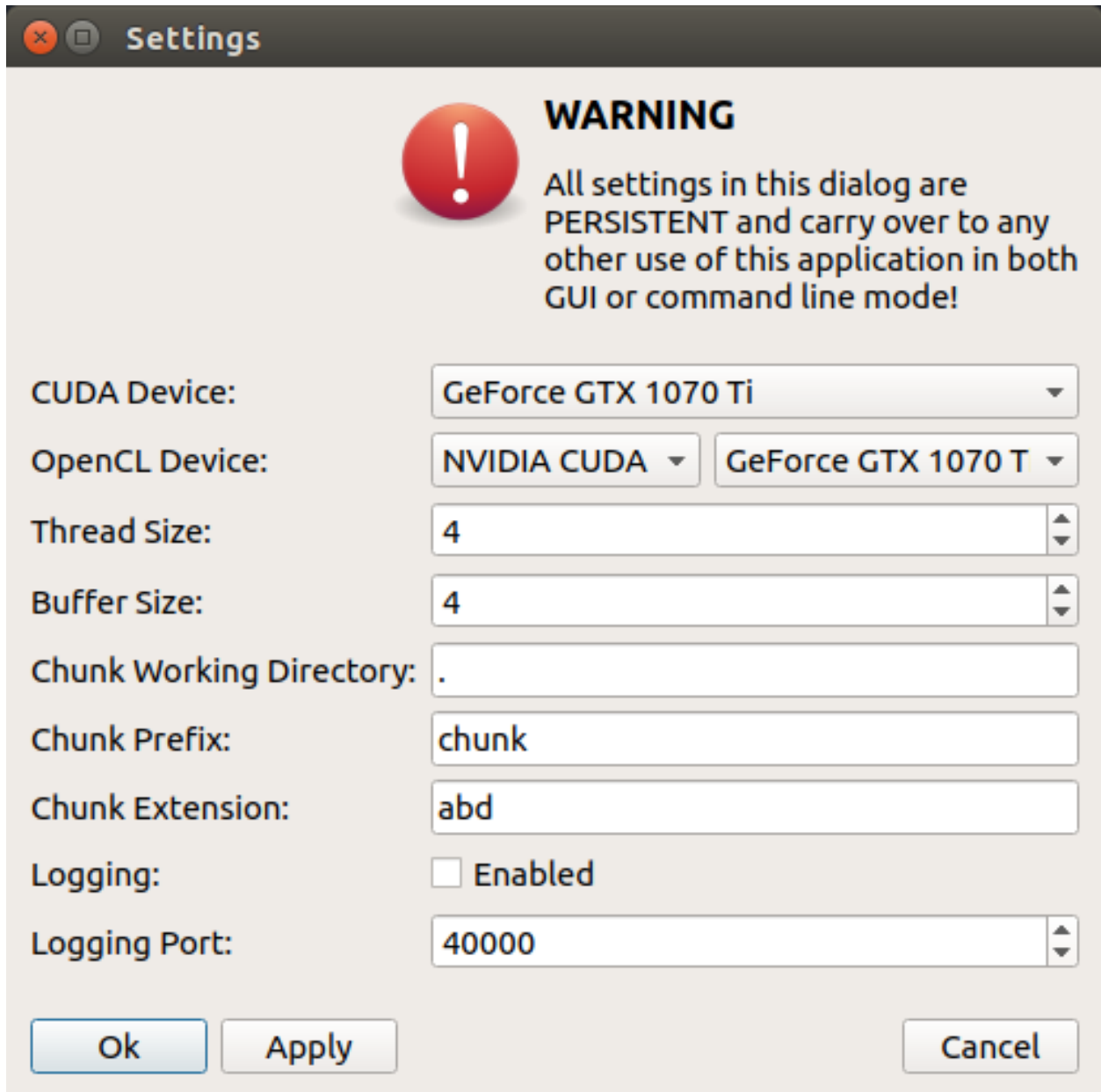
Note: Please see the description of each setting in the *Global Settings* section for the *Command-line Usage* above.

6.3.4 Viewing files


To save storage space and speed calculations, KINC maintains its own compressed file formats in binary. Despite their reduced size, these files can become quite large. Therefore, the KINC GUI offers highly responsive viewers for these files. To view any binary file created by KINC, select the appropriate option from the **Open** menu. An example GEM is shown in the following screenshot by selecting **Open > Expression Matrix**

The similarity (or correlation) matrix can be viewed via the menu **Open > Correlation Matrix** and an example is shown below.

Notice, the correlation matrix is sparse in that many values are missing. This is because KINC was instructed to only retain correlation values above an absolute value of 0.5.

A screenshot of a 'Settings' dialog box. The title bar is dark grey with standard window controls (close, maximize, minimize) and the text 'Settings'. The main area has a light beige background. At the top, there is a red circular warning icon with a white exclamation mark, followed by the word 'WARNING' in bold. Below this, a text block states: 'All settings in this dialog are PERSISTENT and carry over to any other use of this application in both GUI or command line mode!'. The settings are organized into two columns. The left column contains labels: 'CUDA Device:', 'OpenCL Device:', 'Thread Size:', 'Buffer Size:', 'Chunk Working Directory:', 'Chunk Prefix:', 'Chunk Extension:', 'Logging:', and 'Logging Port:'. The right column contains the corresponding input fields: a dropdown menu for 'CUDA Device' showing 'GeForce GTX 1070 Ti', two dropdown menus for 'OpenCL Device' showing 'NVIDIA CUDA' and 'GeForce GTX 1070 T', spinners for 'Thread Size' and 'Buffer Size' both set to '4', text boxes for 'Chunk Working Directory' (containing '.') and 'Chunk Prefix' (containing 'chunk'), a text box for 'Chunk Extension' (containing 'abd'), a checkbox for 'Logging' which is unchecked with the text 'Enabled' next to it, and a spinner for 'Logging Port' set to '40000'. At the bottom, there are three buttons: 'Ok', 'Apply', and 'Cancel'.

Settings

 **WARNING**

All settings in this dialog are PERSISTENT and carry over to any other use of this application in both GUI or command line mode!

CUDA Device: GeForce GTX 1070 Ti

OpenCL Device: NVIDIA CUDA GeForce GTX 1070 T

Thread Size: 4

Buffer Size: 4

Chunk Working Directory: .

Chunk Prefix: chunk

Chunk Extension: abd

Logging: ☐ Enabled

Logging Port: 40000

Ok Apply Cancel

	SRX1424112	SRX1423984	SRX1424118	SRX1424228	SRX1424223	SRX1424228
LOC_Os01g01010	5.6533	6.15241	5.32262	6.38316	4.83054	2.38
LOC_Os01g01019	0.424653	0.432614	0.183151	0.178544	0.237002	0.18
LOC_Os01g01030	1.16978	1.40972	0.780901	1.10462	1.31016	0.72
LOC_Os01g01040	5.15391	7.10479	4.32029	5.66327	5.71282	3.68
LOC_Os01g01050	6.704	6.39867	6.11786	8.21079	4.4333	2.63
LOC_Os01g01060	8.86948	5.09882	12.0833	7.92473	12.489	4.23
LOC_Os01g01070	2.44614	3.30445	1.34841	1.69665	1.55513	0.85
LOC_Os01g01080	9.86193	8.69385	7.58377	13.655	9.74	5.75
LOC_Os01g01115	0.225003	0.121884	0.367448	0.318687	0.204677	0.05
LOC_Os01g01120	27.0034	32.1353	30.1751	23.9839	16.5488	13.2
LOC_Os01g01130	3.66206	5.10906	3.39933	2.62963	3.12048	3.76
LOC_Os01g01150	10.7391	13.2144	8.9607	10.9873	9.85163	8.11
LOC_Os01g01160	10.4827	13.3022	9.60305	9.85486	8.47855	11.2
LOC_Os01g01170	0.317392	0.128502	0.103509	0.045082	nan	nan
LOC_Os01g01190	0.354744	0.230538	0.048953	0.255882	0.199468	0.12
LOC_Os01g01280	4.29077	7.32296	4.93932	4.70464	3.92328	8.35
LOC_Os01g01290	0.269657	0.16234	0.271711	0.155063	0.136164	0.16
LOC_Os01g01295	2.37233	2.81038	1.71826	2.18932	2.42141	3.63
LOC_Os01g01302	19.9874	23.6013	21.3035	20.4206	16.8593	13.8
LOC_Os01g01307	7.54399	7.0529	5.22183	6.04106	6.52424	4.52
LOC_Os01g01312	3.21719	4.6244	3.14901	3.58705	3.12137	3.35

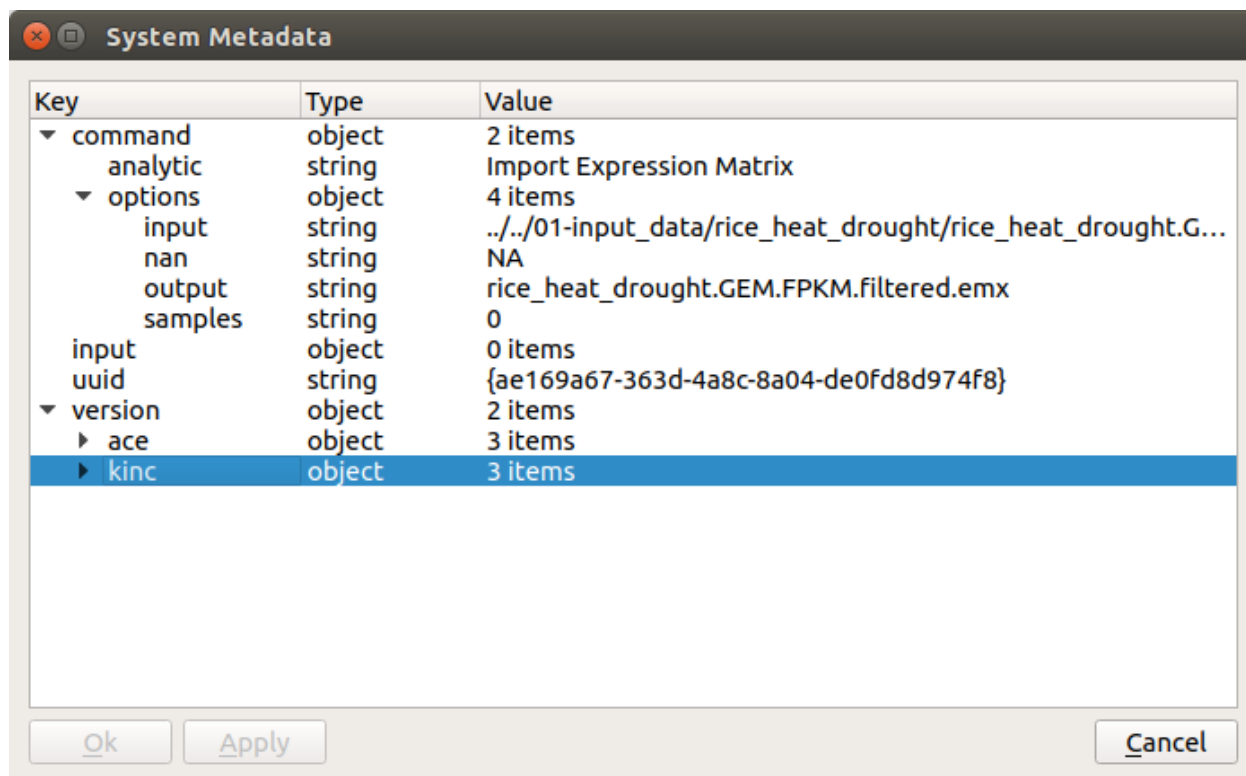
Help Debugger						
	DC_Os01g0101	DC_Os01g0101	DC_Os01g0101	DC_Os01g0101	DC_Os01g0101	DC_Os01g0101
LOC_Os01g01010					0.702695	
LOC_Os01g01019						
LOC_Os01g01030						
LOC_Os01g01040						
LOC_Os01g01050	0.702695					
LOC_Os01g01060						
LOC_Os01g01070				0.516336		
LOC_Os01g01080				0.527341		
LOC_Os01g01115						
LOC_Os01g01120						
LOC_Os01g01130						
LOC_Os01g01150	0.623729			0.598587	0.574401	
LOC_Os01g01160						
LOC_Os01g01170						
LOC_Os01g01190						
LOC_Os01g01280						
LOC_Os01g01290						
LOC_Os01g01295						
LOC_Os01g01302	0.620643					0.52
LOC_Os01g01307	0.624319			0.519841		
LOC_Os01g01312						

6.3.5 Accessing Metadata

KINC strives to ensure reproducibility of results by maintaining system and user metadata within each file. You can access metadata via the GUI for viewing. System meta data maintains a complete provenance for how the file was created, and is immutable. User metadata consists of information about the run of the analytic.

Viewing System Metadata

To view the system meta data for any KINC file, you must first open the file via the `Open` menu. In the window that appears (as seen in the previous figures), a `File` menu is present. Selecting **File > System Metadata** will provide a new window with a clickable tree view of the system metadata. The following view is of the System metadata for the same file shown in the command-line example above.



The screenshot shows a window titled "System Metadata" with a tree view of system metadata. The tree has a root node "command" which is expanded, showing "analytic" and "options". "analytic" is expanded, showing "input", "nan", "output", and "samples". "input" is expanded, showing "input" and "uuid". "version" is expanded, showing "ace" and "kinc". The "kinc" node is selected and highlighted in blue.

Key	Type	Value
command	object	2 items
analytic	string	Import Expression Matrix
options	object	4 items
input	string	../01-input_data/rice_heat_drought/rice_heat_drought.G...
nan	string	NA
output	string	rice_heat_drought.GEM.FPKM.filtered.emx
samples	string	0
input	object	0 items
uuid	string	{ae169a67-363d-4a8c-8a04-de0fd8d974f8}
version	object	2 items
ace	object	3 items
kinc	object	3 items

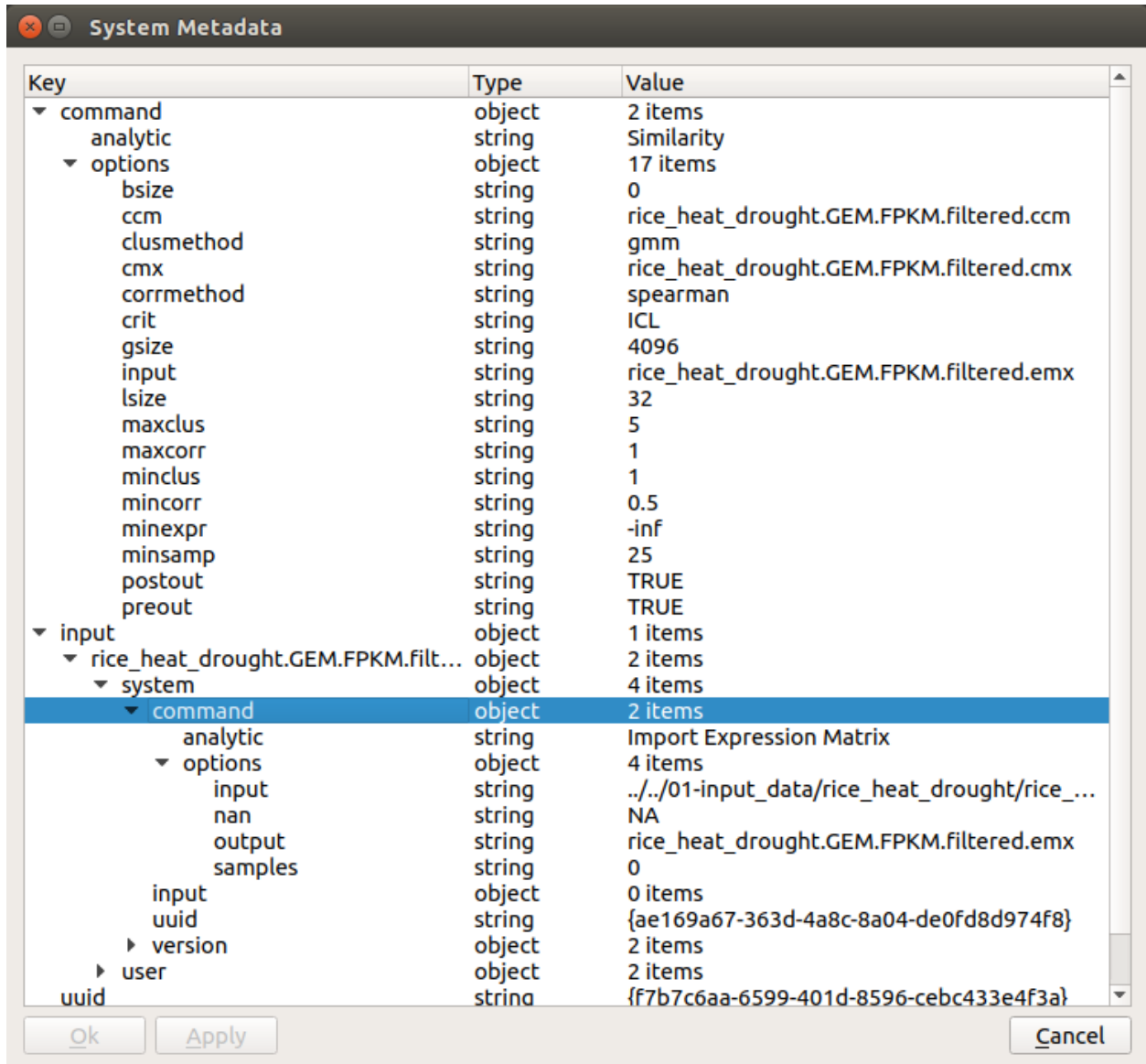
At the bottom of the window are three buttons: "Ok", "Apply", and "Cancel".

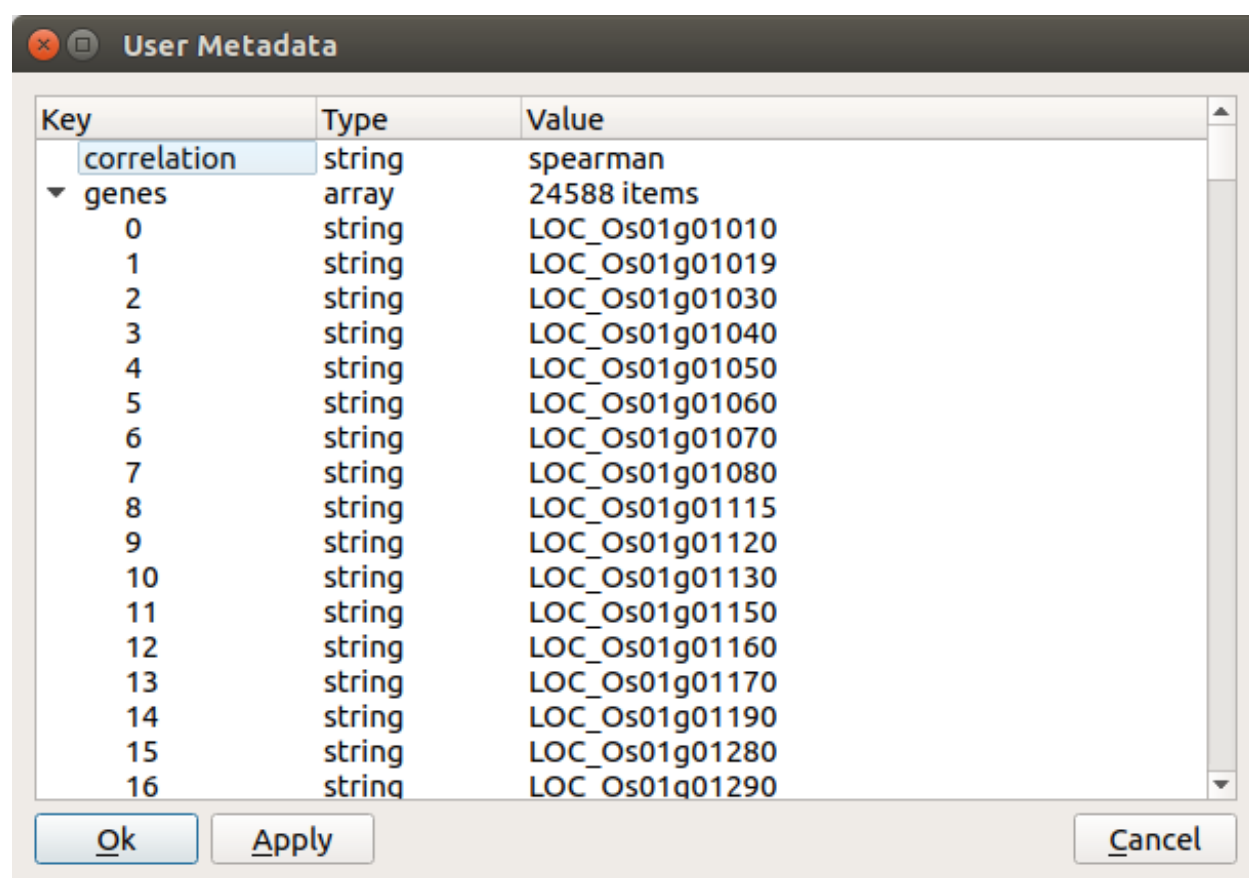
The metadata provides the exact command-line and arguments that were used to produce the file, as well as a unique UUID for the file and the versions of the ACE and KINC that were used to produce the file.

As KINC files are used in other functions, the system metadata is preserved. Therefore the complete provenance from beginning to end for creation of the file is maintained. The following view of system metadata is from a correlation matrix (.cmx extension) file that used the expression matrix file as input. Notice it has the exact command-line arguments for the `similarity` function, but also includes the system metadata for all of the input files that it used, including the expression matrix.

Retrieving User Metadata

User metadata can be retrieved by selecting the **File > User Metadata** menu item in the file viewer window. An example of the user metadata from a correlation matrix file:





6.4 Using KINC with Docker


KINC can be installed and run on both a stand-alone workstation or via an HPC cluster. However, sometimes it is not possible to install the dependencies required by KINC. Therefore, a Docker image and a Nextflow workflow are provided to help ease use when installation proves difficult.

This solution does require installation of **Docker** which does require administrative (i.e. root) access to the machine. You must also follow the instructions on the *nvidia-docker* <<https://github.com/NVIDIA/nvidia-docker>>_ site to make this work.

The KINC docker image comes pre-installed with all dependencies. The Dockerfile for KINC is available in the KINC Github repository, and Docker images are maintained on DockerHub under *systemsgenetics/kinc*. This method currently does not support the GUI version of KINC.


6.4.1 Interactive Mode

To use KINC in an interactive Docker container **with GPUs** execute the following:

```
docker run --gpus all --rm -it --net=host -v ${PWD}:/workspace -u $(id -u ${USER}) 
↪systemsgenetics/kinc:3.4.2-gpu /bin/bash
```

The command above will provide access to the terminal inside of the *systemsgenetics/kinc:3.4.2-gpu* image.

To use KINC in an interactive Docker container **without GPUs** execute the following:

```
docker run --gpus all --rm -it --net=host -v ${PWD}:/workspace -u $(id -u ${USER}) 
↪systemsgenetics/kinc:3.4.2-cpu /bin/bash
```

The above command uses the image *systemsgenetics/kinc:3.4.2-cpu* (note the “-cpu” suffix).

The following describes the meaning of each argument in the commands above:


- *--gpus all*: ensures that the NVidia CUDA libraries are present in the image. This is needed even if using only CPUs.
- *--rm*: will cause the container to be cleaned up after you exit.
- *-it*: tells Docker to run in interactive mode with a terminal
- *--net=host*: exposes network ports in the image to your local machine. This is needed to use the Docker image for the 3D KINC viewer.
- *-v \${PWD}:/workspace*: adds the current directory on the local machine as a filesystem in the image accessible via */workspace*. This allows you to work with files on your local machine inside of the image.
- *-u \$(id -u \${USER})*: logs you in as your local user account so that any files created within directories mounted by the image are saved with your user account.

Once inside the KINC Docker image you can execute commands such as the following:

```
> nvidia-smi
> kinc settings
```

6.4.2 Test The Example Data

To test the docker image using the example data that comes with KINC. Run the following inside of the KINC source directory on your local machine:


```
docker run --gpus all --rm -it --net=host -v ${PWD}:/workspace -u $(id -u ${USER}) 
↪systemsgenetics/kinc:3.4.2-cpu /bin/bash
```

Next run the following commands inside of the container:

```
cd /workspace/example
./kinc-gmm-run.sh
```

6.4.3 Using the 3D Visualization Tool

To use the 3D visualization tool to explore a KINC created network first start an interactive session in the directory where the network file(s) are stored:

```
docker run --gpus all --rm -it --net=host -v ${PWD}:/workspace -u $(id -u ${USER}) 
↪systemsgenetics/kinc:3.4.2-cpu /bin/bash
```

Then run the *kinc-3d-viewer.py*. For example, if the example data is already fully processed using the steps in the previous section you can view the results with the viewer with the following commands.

```
cd /workspace/example/results-kinc-gmm-run
kinc-3d-viewer.py \
  --net "PRJNA301554.slim.GEM.log2.paf-th0.00-ple-3-rsq0.30-filtered-th_ranked.csGCN.
↪txt" \
  --emx "../data/PRJNA301554.slim.GEM.log2.txt" \
  --amx "../data/PRJNA301554.slim.annotations.txt"
```

The first time the viewer is run you will see output about creating 2D and 3D layouts. Once completed you will see a line of text similar to the following:

```
* Running on http://127.0.0.1:8050/ (Press CTRL+C to quit)
```

Copy the URL into a web browser and view the network.

6.5 Automating KINC with Nextflow

Once you are familiar with KINC and wish to automate the full workflow, you can use the [KINC-nf](#) nextflow pipeline, which can run the full KINC workflow on nearly any computing environment with very little setup required. Consult the KINC-nf repository on Github for instructions.

Note: Before you use the KINC-nf workflow to automate the full process, it is recommended to be fully familiar with KINC and all of its functions.

How to Create a Network

This section describes the two approaches for network construction as well as some general considerations. For the examples shown below we will use an input GEM that is derived from the SRA Project with ID [PRJNA301554](#). This dataset consists of 475 RNA-seq Illumina samples of rice grown under control, heat, drought, heat recovery and drought recover conditions. It measures expression across four different genotypes of rice (from two subspecies) over a series of time points. This dataset was selected because of its large size and multiple variables.

For the purposes of this tutorial, a sample GEM file, containing 60 samples from the PRJNA301554 experiment is used. This file is named `PRJNA301554.slim.GEM.log2.txt` and is found in the `example` directory of the KINC source code.

Note: If you cannot find the `example` directory for KINC on your local machine you can download the files from the [KINC Github repository](#).

The example file has been limited to 60 samples rather than 475 to ensure that KINC executes quickly on a single workstation for the purposes of this tutorial. These 60 samples consist of 30 randomly selected heat treatment samples and 30 randomly selected control samples. The tool [GSForge](#) was used to find the genes that are most predictive of the experimental conditions. This resulted in a small set of 1,167 genes, and these are present in the example GEM file. The gene expression levels (in FPKMs) for these 60 samples were quantified using [GEMmaker](#).

7.1 Before Getting Started

7.1.1 Traditional or GMM Approach?

Before proceeding you should identify if a traditional or GMM network is most appropriate. The following considerations can help with that decision.

Traditional Approach

Advantages

- Executes very fast.
- Biological signal can be found at higher correlation values and when sample bias is skewed towards the question of interest

Disadvantages

- Includes false edges due to improper application of correlation tests, and misses lower-correlated true edges due to excess noise.

GMM Approach**Advantages**

- Limits correlation test bias by using GMMs to ensure test assumptions are met.
- Can identify condition-specific pairwise expression.
- Reduces the effect of sample bias when multiple conditions are present in the input GEM dataset.
- Can execute very fast on a single machine for small GEMs.

Disadvantages

- May require access to HPC with multiple GPUs for very large GEMs.
- May not be useful if the sample metadata does not have qualitative (categorical) conditions.

7.1.2 How Many Samples are Needed?

Networks can be created with very few samples if need be, but the power of the network will diminish greatly. For traditional networks, you can manually perform a power analysis before network construction to identify what correlation value (i.e. effect size) you must not go below in thresholding in order to limit false edges (assuming correlation assumptions are met, which they are not in the traditional approach). The `pwr.r.test` function of the statistical programming language R can do this, and there are [online calculators](#) as well.

For example, the minimum number of samples required to meet the criteria of for a significance value of 0.001, a power value of 0.8 and a minimum correlation threshold of 0.5 is 60 samples. If we raise the minimum threshold to 0.7 we need at least 21 samples. Only 11 samples are needed for a threshold limit of 0.9. Thus, if we only had 11 samples we should not allow a correlation threshold below 0.9.

If you are using the GMM approach and you wish to find condition-specific subgraphs for a qualitative condition, such as for genes underlying response to a treatment (e.g.heat, drought, control, etc.) you must ensure that you have sufficient samples for each category. Suppose you only had 10 samples per treatment, you would expect to find clusters of approximately size 10 for each treatment, and this would require a minimum correlation threshold of 0.9. You can remove edges whose correlation dips below the limit using the `corrpwr` function. You can set the minimum cluster size when executing the `similarity` function.

Note: The number of samples will dictate the quantity and size of the final network. With few samples there is little chance of finding weakly correlated, but perhaps meaningful edges.

7.1.3 Do Replicates Matter?

Unlike with DGE analysis, where multiple replicates are necessary to establish a statistical difference between two conditions, for GCNs a relationship between genes can be established using all data for that gene. Therefore, It is

not clear what contribution replicates make in the network. Technical replicates are probably not useful. Biological replicates may be useful but the benefit or bias from using replicates is not yet known.

7.1.4 Which Correlation Method to Choose?

KINC currently provides two correlation methods: Pearson and Spearman. Pearson is meant for linear relationships with no outliers that have equal variance. Spearman is less affected by outliers and non-linearity so long as the relationship is monotonically increasing. It is therefore recommended to use Spearman as not all relationships are linear. However, Spearman correlation is less effective with small samples. Empirically, Spearman tends to suffer when sample sizes dip below 20 to 30. If you wish to identify condition-specific edges where each category (e.g. heat, drought, control) has fewer than 20 to 30 samples you should consider using Pearson.

7.2 Traditional Approach

You can construct a traditional network on a stand-alone workstation using either `kinc` or `qkinc`. Using the 60-sample example rice dataset described above, the following steps show how to create a traditional network using the command-line. The arguments shown in the command-line examples below correspond directly to fields in the KINC GUI.

7.2.1 Step 1: Import the GEM

The first step is to import the GEM into a binary format suitable for KINC. The `import-emx` function of KINC does this:

```
kinc run import-emx \
  --input "PRJNA301554.slim.GEM.log2.txt" \
  --output "PRJNA301554.slim.GEM.log2.emx" \
  --nan "NA" \
  --samples 0
```

In the example code above the GEM file is provided to the `--input` argument and the name of an output EMX file is provided using the `--output` argument. In the example above, the `--nan` argument indicates that the file uses "NA" to represent missing values. This value should be set to whatever indicates missing values. This could be "0", "0.", "-Inf", etc. and the GEM file has a header describing each column so the number of samples provided to the `--samples` argument is set to 0. If the file did not have a header the number of samples would need to be provided.

7.2.2 Step 2: Perform Correlation Analysis

Construction of a similarity matrix (or correlation matrix) is the second step. Here KINC performs pairwise comparison of every gene with every other gene using either Spearman or Pearson correlation. The `similarity` function of KINC does this:

```
kinc run similarity \
  --input "PRJNA301554.slim.GEM.log2.emx" \
  --ccm "PRJNA301554.slim.GEM.log2.traditional.ccm" \
  --cmx "PRJNA301554.slim.GEM.log2.traditional.cmx" \
  --clusmethod "none" \
  --corrmethod "spearman" \
  --minsamp 30 \
  --minexpr -inf \
```

(continues on next page)

(continued from previous page)

```
--mincorr 0.5 \  
--maxcorr 1
```

Here the EMX file created in the first step is provided using the `--emx` argument and the names of two output files are provided using the `--cmx` and `--ccm` arguments. These are the correlation matrix and clustering matrix respectively. Because we are using the traditional approach, the `--clusmethod` argument is set to "none". The correlation method is set to use Spearman, and the minimum number of samples required to perform correlation is set to 30 using the `--minsamp` argument. Any gene pairs where one gene has fewer than `--minsamp` samples will be excluded. This will exclude genes that have missing values in samples that causes the number of samples to dip below this level. The `--minsamp` argument should be set equal to or lower than the number of samples present in the origin GEM input file and higher than an expected level of missigness (e.g. 10% missing values allowed). The `--minexp` argument is set to negative infinity (`-inf`) to indicate there is no limit on the minimum expression value. If we wanted to exclude samples whose log2 expression values dipped below 0.2, for instance, we could do so with this argument. To keep the output files relatively small, we will exclude all correlation values below 0.5 using the `--mincorr` argument. Sometimes errors occur in data collection or quantification yielding high numbers of perfectly correlated genes! We can limit that by excluding perfectly correlated genes by lowering the `--maxcorr` argument. In practice we leave this as 1 for the first time we create the network, if we fail to find a proper threshold in a later step then one cause may be large numbers of perfectly correlated genes.

7.2.3 Step 3: Thresholding

There are four ways KINC can determine a threshold for a network: power-law, Random Matrix Theory (RMT), condition-specific and *ad hoc*. RMT is the recommended approach for traditional networks.

Method 1: Using RMT to Threshold

The following command-line provides an example for RMT thresholding of the example 475-rice sample data:

Note: RMT works best for traditional networks.

```
kinc run rmt \  
  --input "PRJNA301554.slim.GEM.log2.traditional.cmx" \  
  --log "PRJNA301554.slim.GEM.log2.traditional.rmt.log" \  
  --tstart "0.95" \  
  --tstep "0.001" \  
  --tstop "0.5" \  
  --threads 1 \  
  --epsilon 1e-6 \  
  --mineigens 50 \  
  --spline true \  
  --minpace 10 \  
  --maxpace 40 \  
  --bins 60
```

The above command provides the correlation matrix (CMX) using the `--input` argument, and the name of a log file, using the `--log` argument where the results of chi-square testing is stored. The RMT method will successively walk through all correlation values, in decreasing order from `--tstart` to `--tstop`, using a step of `--tstep`, and builds a new similarity matrix to test if the Nearest Neighbor Spacing Distribution (NNSD) of the Eigenvalues of that matrix appears Poisson. A spline curve is fit to the NNSD if the `--spline` argument is `TRUE` (recommended) and random points along the line are selected to determine if the distribution appears Poisson. This random selection will occur repeatedly by selecting a random set of `--minpace` numbers and increasing that on successive iterations

to `--maxpace`. A Chi-square test is performed for each of these random selections and the result is averaged for each correlation value. The `--bins` is the number of bins in the NNSD histogram and `1 - bins` indicates how many degrees of freedom the Chi-square test will have. In practice, a Chi-square value of 100 indicates that the correlation value begins to not look Poisson. The RMT approach will continue after seeing a Chi-square value of 100 until it sees one at the 200 at which point it stops. It seeks past 100 to ensure it does not get trapped in a local minimum.

Note: It is best to leave all options as default unless you know how to tweak the RMT process.

Once completed, you can determine the best threshold for the network by opening the logfile specified by the `--log` argument, and looking at the end of the file. The threshold is listed on the last line of the file and should be used for extracting the network in step 4.

If the input GEM is especially noisy, the RMT method will fail to find a threshold. As it continues to search through decreasing correlation values, the time required to generate the eigenvalues dramatically increases and it may appear that RMT never completes. To determine if this is the case, examine the log file. If you see the average correlation beyond 200 then this has occurred. See the [Troubleshooting](#) section to explore alternative methods.

Method 2: Using the Power-law Threshold

The Power-law function tests to see if the network, at successively decreasing correlation values follows a power-law which is a property of scale-free network. The power-law threshold can be used as an alternative to RMT when it fails to find a solution. The following example uses the power-law threshold for the example 475-rice sample data:

```
kinc run powerlaw \
  --input "PRJNA301554.slim.GEM.log2.traditional.cmx" \
  --log "PRJNA301554.slim.GEM.log2.traditional.powerlaw.log" \
  --tstart 0.99 \
  --tstep 0.01 \
  --tstop 0.5
```

Here the correlation matrix (CMX) file is provided as well as a log file where details about the analysis are stored. The `--tstart` argument sets the starting correlation value and power-law calculations continue until the `--tstop` value is reached.

If function fails to find an threshold then see the [Troubleshooting](#) section to explore alternative methods.

Warning: While the power-law threshold is useful to help identify scale-free behavior, it does not that the network is modular and hierarchical.

Method 3: Applying a Condition-Specific Filter

The condition-specific thresholding approach uses an annotation matrix that contains metadata about the samples such as the experimental conditions or phenotypes. The approach to perform condition-specific thresholding is the same as for the GMM approach. Please refer to the `csfilter-reference-label` section for details about using condition-specific filters for either traditional or GMM networks.

Warning: Condition-specific thresholding only works with traditional networks when the metadata in the annotation matrix is quantitative.

Method 2: Using an *Ad Hoc* Approach

An *ad hoc* threshold does not use an analytical approach to determine a threshold. Instead, the researcher selects a reasonable threshold. For example, this could be the minimum correlation that selects the top 1000 relationships, or yields a network that has desired size or communities. These types of thresholding approaches have been used for peer-reviewed published networks but users should be cautious when using this approach.

7.2.4 Step 4: Extracting the Network File

How ever you have chosen to threshold the network, either with RMT or Power-law, or some *ad-hoc* approach, you will have a minimum correlation value. This value can be used to extract any pairwise comparison between genes in the correlation matrix file (CMX) that are above the absolute value of the minimum correlation. These become edges in the final network. The `extract` function of KINC will do this:

```
kinc run extract \  
  --emx "PRJNA301554.slim.GEM.log2.emx" \  
  --ccm "PRJNA301554.slim.GEM.log2.traditional.ccm" \  
  --cmx "PRJNA301554.slim.GEM.log2.traditional.cmx" \  
  --format "tidy" \  
  --output "PRJNA301554.slim.GEM.log2.traditional.paf-th0.826002-gcn.txt" \  
  --mincorr 0.826002 \  
  --maxcorr 1
```

As in previous steps, the `--emx`, `--cmx` and `--ccm` arguments provide the expression matrix, correlation and clustering matrices. The threshold is provided to the `--mincorr` argument. Additionally, if you would like to exclude high correlations (such as perfect correlations), you can do so with the `--maxcorr` argument. You should only need to change the `--maxcorr` argument if it was determined that there is error in the data resulting in an inordinate number of high correlations. The `--format` argument can be `text`, `minimal` or `graphml`. The `text` format currently contains the most data. It is easily imported into Cytoscape or R for other analyses and visualizations. The `minimal` format simply contains the list of edges with only the two genes and the correlation value. The `graphml` format provides the same information as the `minimal` format but using the [GraphML](#) file format.

See the [Plain-text Output Files](#) section for specific details about these files.

7.3 GMM approach

Here we perform network construction using the Gaussian Mixture Model (GMM) approach. With this approach, each pair-wise comparison of every two genes undergoes a cluster identification analysis using GMMs. This approach can identify clusters, or groups, of samples that have similar but distinct ranges of expression. The underlying hypothesis is that when clusters appear, they represent condition-specific gene expression. Clusters that are identified in gene pairs are correlated independently and each cluster has the potential to become a separate edge in the network. Because we know the samples that are present in each cluster, KINC uses a hypergeometric test to compare categorical data about samples with cluster membership, and regression analysis to compare qualitative and ordinal data. Condition-specific thresholding can be performed on the p -values and r -squared values of those test to generate condition-specific subgraphs.

Note: The GMMs approach requires a tab-delimited annotation matrix file (AMX) that contains metadata about samples where columns are feature that contain experimental condition information or phenotype data.

7.3.1 Step 1: Import the GEM

```
kinc run import-emx \
  --input "PRJNA301554.slim.GEM.log2.txt" \
  --output "PRJNA301554.slim.GEM.log2.emx" \
  --nan "NA" \
  --samples 0
```

In the code above the GEM file is provided to the `import-emx` function and the name of an output EMX file is provided. The file uses “NA” to indicate missing values and it has a header so the number of samples is set to .

7.3.2 Step 2: Perform GMM + Correlation Analysis

The second step is to use GMM to identify clusters and then perform correlation analysis on each cluster.

```
kinc run similarity \
  --input "PRJNA301554.slim.GEM.log2.emx" \
  --ccm "PRJNA301554.slim.GEM.log2.ccm" \
  --cmx "PRJNA301554.slim.GEM.log2.cmx" \
  --clusmethod "gmm" \
  --corrmethod "spearman" \
  --minexpr -inf \
  --minsamp 25 \
  --minclus 1 \
  --maxclus 5 \
  --crit "ICL" \
  --preout TRUE \
  --postout TRUE \
  --mincorr 0 \
  --maxcorr 1
```

Here the EMX file created in the first step is provided, and the names of the two output (CCM and CMX) files are provided. Because we are using the GMM approach, the `--clusmethod` argument is set to "gmm". The correlation method is set to use Spearman. Other argument specific to the GMM approach include `--crit`, `--maxclus`, `--minclus`, `--preout`, and `--postout`. These have the following meaning:

- `--crit`: This is the criterion to select a clustering model. This should remain as ICL unless a higher number of modules per pair is desired and can be set to ‘BIC’.
- `--minclus`: The minimum number of clusters that can be found per gene pair. Unless you are specifically looking for genes with multi-modal expression this should remain s 1.
- `--maxclus`: The maximum number of clusters that can be found per gene pair.
- `--preout`: Set to TRUE to turn on removal of outliers prior to GMM clustering. FALSE otherwise.
- `--postout`: Set to TRUE to remove outliers that may be present in GMM clusters. FALSE otherwise.

The `--minexp` argument is set to negative infinity (`-inf`) to indicate there is no limit on the minimum expression value. If we wanted to exclude samples whose log2 expression values dipped below 0.2, for instance, we could do so. To keep the output files relatively small, we will exclude all correlation values below 0.5 using the `--mincorr` argument.

Sometimes errors occur in data collection or quantification yielding high numbers of perfectly correlated genes! We can limit that by excluding perfectly correlated genes by lowering the `--maxcorr` argument. In practice we leave this as 1 for the first time we create the network.

7.3.3 Step 3: Filter Low-Powered Edges

As discussed in the *How Many Samples are Needed?* section above, the power of a correlation analysis is dependent on the number of samples in the test. Unlike the traditional approach, where a power analysis can indicate the minimum correlation threshold below which you should not drop, a power-analysis for the GMM approach must be applied to each cluster separately. The `corrpower` function does this and removes underpowered clusters from the matrices. For example:

```
kinc run corrpower \  
  --ccm-in "PRJNA301554.slim.GEM.log2.ccm" \  
  --cmx-in "PRJNA301554.slim.GEM.log2.cmx" \  
  --ccm-out "PRJNA301554.slim.GEM.log2.paf.ccm" \  
  --cmx-out "PRJNA301554.slim.GEM.log2.paf.cmx" \  
  --alpha 0.001 \  
  --power 0.8
```

As shown above, the power and significance criteria are set with the `--power` and `--alpha` arguments respectively. An alpha setting of 0.001 indicates that we want to limit the Type I error (false positives) to a significance level of 0.001. The Power uses the formula $1 - \text{Beta}$ where *Beta* is the probability of a Type II error (false negative) occurring. A `--power` setting of 0.8 indicates that we are comfortable with a 20% false negative rate. There is no rule for how to set these. Set them to the levels of noise you are comfortable with.

Note: Remember, to find edges in the network associated with categorical features, you must have enough samples with the given category in order to find a cluster and then to have sufficient power. The `--minsamp` `` argument in the ``similarity step sets the smallest allowable cluster size.

7.3.4 Step 4: Condition-Specific Filtering

Condition-specific filtering is performed using the `cond-test` function of KINC. It requires an annotation matrix containing metadata about the RNA-seq samples. It performs a hypergeometric test for categorical features and linear regression analysis for quantitative features that assigns *p*-values and R² values, as appropriate, to each edge in the network. The following shows an example:

```
kinc run cond-test \  
  --emx "PRJNA301554.slim.GEM.log2.emx" \  
  --ccm "PRJNA301554.slim.GEM.log2.paf.ccm" \  
  --cmx "PRJNA301554.slim.GEM.log2.paf.cmx" \  
  --amx "PRJNA301554.slim.annotations.txt" \  
  --output "PRJNA301554.slim.GEM.log2.paf.csm" \  
  --feat-tests "Subspecies,Treatment,GTAbbr,Duration" \  
  --feat-types "Duration:quantitative"
```

Here, the `--emx`, `--ccm`, and `--cmx` arguments provide the usual expression matrix, cluster matrix and correlation matrix respectively. The `--amx` argument specifies the *Sample Annotation Matrix (AMX)*. The name of new condition-specific matrix, that will contain the results of the tests is set using the `--output` argument.

Finally, it may not be desired to test all of the metadata features (i.e. columns) from the annotation matrix. Using the `feat-tests` argument you can specify a comma-separated list (without spaces) of the names of the columns in the annotation matrix file that should be tested. These can be either categorical, quantitative or ordinal. KINC will do its best to determine the top of data in each column, but you can override the type using the `--feat-types` argument and specifying the type by separating with a colon.

7.3.5 Step 5: Extract Condition-Specific Subgraphs

When using the GMM approach, the goal is to identify condition-specific subgraphs. These are subsets of a larger “unseen” network that are specific to a given condition. As with the traditional approach, the `extract` function of KINC will do this:

```
p="1e-3"
r2="0.30"
th="0.00"
kinc run extract \
  --emx "PRJNA301554.slim.GEM.log2.emx" \
  --ccm "PRJNA301554.slim.GEM.log2.paf.ccm" \
  --cmx "PRJNA301554.slim.GEM.log2.paf.cmx" \
  --csm "PRJNA301554.slim.GEM.log2.paf.csm" \
  --format "tidy" \
  --output "PRJNA301554.slim.GEM.log2.paf-th${th}-p${p}-rsqr${r2}.txt" \
  --mincorr $th \
  --maxcorr 1 \
  --filter-pvalue $p \
  --filter-rsquare $r2
```

As in previous steps, the `--emx`, `--cmx`, `--ccm` and `--csm` arguments provide the expression matrix, correlation, clustering matrix and the new condition-specific matrix. A threshold is provided to the `--mincorr` argument typically as a lower-bound. No edges with absolute correlation values below this value will be extracted. Additionally, if you would like to exclude high correlations (such as perfect correlations), you can do so with the `--maxcorr` argument. You should only need to change the `--maxcorr` argument if it was determined that there is error in the data resulting from an inordinate number of high correlations. In the example above the `--mincorr` is set at 0.5. This is quite low by traditional standards but the following filtering and thresholding steps support exploration of edges at such a low correlation.

To limit the size of the condition-specific subgraphs you should then set the `--filter-pvalue` and `--filter-rsquare` values to lower-bounds for significant p-values and meaningful R-square values from test. The R-square values are only present for quantitative features where the regression test was performed. The p-value in this case indicates how well the data follows a trend and the r-square indicates how much of the variation the trend line accounts for. Ideally, low p-values and high r-square are desired. However, there are no rules for the best setting, but choose settings that provide a significance level you are comfortable with.

Finally, the `--format` argument can be `tidy`, `text`, `minimal` or `graphml`. The `tidy` format is recommended for use by later steps. The [GraphML](#) version is larger in size and in an XML format compatible with other graph tools. The `tidy`, `text` and `graphml` formats are easily imported into Cytoscape. The `minimal` format contains the list of edges with only the two genes and the correlation value. See the [Plain-text Output Files](#) section for specific details about these files.

Complex Filtering

For either the `--filter-pvalue` or `--filter-rsquare` you can specify more complex filters in any of the following forms:

1. `[value]`
2. `[class],[value]`
3. `[class],["gt","lt"],[value]`
4. `[class],[label],[value]`
5. `[class],[label],["gt","lt"],[value]`

Where:

- `[value]` is a p-value or r-squared value on which edges should be filtered.
- `[class]` is the name of a condition (i.e. the column header label in the annotation matrix) where any tests performed by the `cond-test` function should be applied.
- `["gt", "lt"]` is either the abbreviation “gt” or “lt” indicating if values “greater than” or “less than” that specified by `[value]` should be extracted.
- `[label]` is set to a category label within the condition class (for categorical data only) to further refine filtering of categorical test results.

If a `[value]` filter is provided (i.e. only a single numeric value), as in the example code above, then the filter applies to all tests that were performed. For example, a filter of `1e-3` indicates that any test performed in the `cond-test` step that has a value less than `1e-3` should be extracted.

If a `[class]`, `[value]` filter is provided then the filter applies to only tests for the given label, and all other tests are ignored. For example. To find edges that are specific to any *Subspecies* with a p-value $< 1\text{-}e3$, the following filter could be supplied: `Subspecies, 1e-3`. If “gt” or “lt” is missing it is implied as “lt” for p-value filters and “gt” for r-squared filters.

If a `[class]`, `["gt", "lt"]`, `[value]` filter is provided then the filter is the same as the `[class]`, `[value]` except that the selection of greater than or less than is explicitly stated.

Finally, the filters, `[class]`, `[label]`, `[value]` and `[class]`, `[label]`, `["gt", "lt"]`, `[value]`, are only applicable to tests where categorical data was used. The latter explicitly provides the “greater than” or “less than” specification. Here the filter specifically expects a given category. For example. To find edges that are specific to only the *Indica* subspecies with a p-value $< 1\text{-}e3$, the following filter could be supplied: `Subspecies, Indica, lt, 1e-3`. If “gt” or “lt” is missing it is implied as “lt” for p-value filters and “gt” for r-squared filters.

Filters can be combined by separating them with two colons: `::`. For example, to find edges that are specific to heat but not heat recovery the following would require a significant p-value for Heat and a non-significant p-value for heat recovery: `Treatment, Heat, 1e-3::Treatment, Heat Recovery, gt, 1-e3`

Note: Filters limit the extracted edge list by finding edges that meet the criteria but do not exclude edges that may be significant for other tests. For example, If the filter `Treatment, Heat, 1e-3` is supplied it will find edges that meet that filter but does not imply the other tests such as a significant *Subspecies* is not also present.

7.3.6 Step 6: Remove Biased Edges

This step must be performed using the `filter-condition-edges.R` R script. It uses **‘KINC.R<<https://github.com/SystemsGenetics/KINC.R>>’** package, an R companion library for KINC. KINC is an actively developed software project and functions are often implemented in R before being moved to the faster C++ based KINC software. *You must install KINC.R prior to using this script.* A false edge can be present under two known conditions:

1. **Lack of differential cluster expression (DCE).** Lack of DCE occurs when GMMs detected a unique cluster in the co-expression between two genes, but the mean expression level of each cluster is not different between the two genes. For the condition-specific edge to be true, it must have different expression within the cluster than without. The script uses a Welch’s Anova test to compare the mean expression of the in- and out-cluster samples for each gene. This test allows for unequal variance.
2. **Unbalanced missing data.** When one gene has more missing expression values than another it biases the co-expression relationship towards a condition if that gene’s expression is condition-specific. The missingness patterns of both genes must be similar. A T-test is used to compare the difference in missingness between the two genes of an edge.

The following example demonstrates how to remove biased edges:

```
kinc-filter-bias.R \  
  --net "PRJNA301554.slim.GEM.log2.paf-th0.00-p1e-3-rsq0.30.txt" \  
  --emx "PRJNA301554.slim.GEM.log2.txt" \  
  --out_prefix "PRJNA301554.slim.GEM.log2.paf-th0.00-p1e-3-rsq0.30"
```

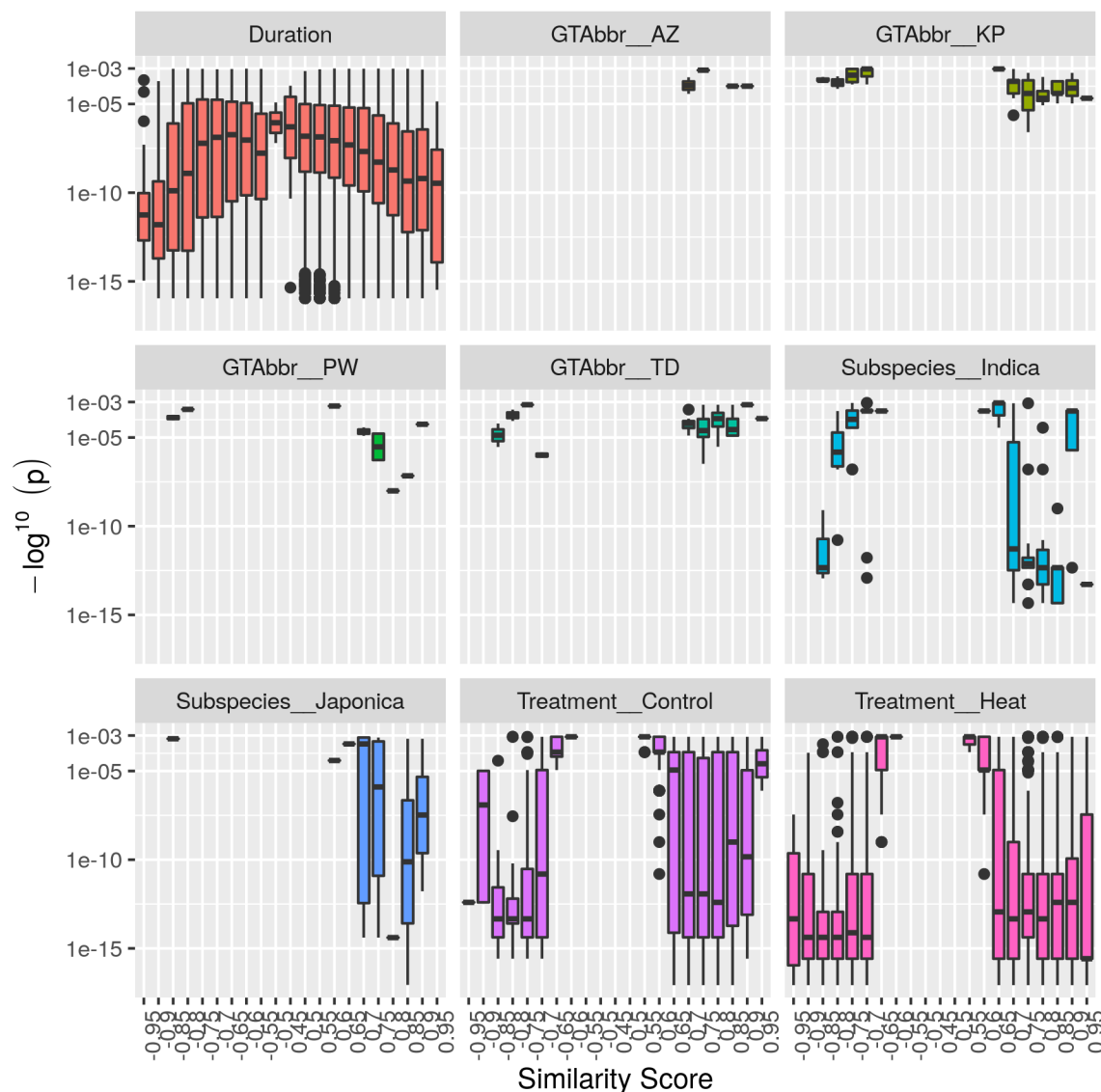
Here the `--net` argument specifies the name of the network created using the `extract` function of KINC. It must be in tidy format. The `--emx` argument specifies the original GEM provided to the `import-emx` function, and the `--out_prefix` provides a name for the filtered output file.

By default the script will use a p -value threshold of $1e-3$ for the Welch's Anova test and 0.1 for the t-test. Edges with a p -value less than $1e-3$ will be kept for the Welch's test and a p -value greater than 0.1 (indicating a difference in missigness can't be detected) for the t-test. You can adjust these thresholds using the `--wa_th` and `--mtt_th` arguments respectively. See the help text (by running the script with no arguments) for other options.

Warning: If the condition-specific network, extracted from KINC is very large (i.e several Gigabytes in size) it may be slow to run this script. The script is multi-threaded and by default will use all but 2 of the available CPU cores to speed up processing.

7.3.7 Step 7: Generate Summary Plots

After filtering of the network, it is useful to explore how the distribution of p -values and $R_{\text{sup:2}}$ values differ between conditions and similarity scores. This helps understand the level of condition-specific response in the network. This can be performed using the `make-summary-plots.R` R script, which also uses KINC.R. The following is an example of a plot generated by this script:



The following is an example to generate the summary plots:

```
kinc-make-plots.R \
  --net "PRJNA301554.slim.GEM.log2.paf-th0.00-p1e-3-rsq0.30-filtered.GCN.txt" \
  --out_prefix "PRJNA301554.slim.GEM.log2.paf-th0.00-p1e-3-rsq0.30-filtered"
```

Here the `--net` argument specifies the name of the network. This should be the network created after Step 6: filtering biased conditional edges. The `--out_prefix` provides the file name prefix for the output images.

7.3.8 Step 8: Threshold the Network by Ranks

In many cases the condition-specific networks can be very large. This is especially true if time-series data is present and if during the `extract` function of KINC a very low minimum similarity score threshold was used (e.g. 0.5). It is not yet clear how many false or true edges remain in the network. Therefore, it is beneficial to perform one last threshold to reduce the size of the network. Here, the similarity score, p -value and R:sup:2 values for each edge are ordered and ranked independently. Then a valuation of each edge, based on the weighted sum of all of the ranks is

calculated. Finally the edges are given a final rank in order (smallest to largest) by their valuation. You can then perform an *ad hoc* filtering by retaining only the top n edges.

To perform this ranking the Rscript `rank-condition-threshold.R` is used. It too uses KINC.R. The following provides an example for filtering the entire network.

```
kinc-filter-rank.R \
  --net "PRJNA301554.slim.GEM.log2.paf-th0.00-ple-3-rsq0.30-filtered.GCN.txt" \
  --out_prefix "PRJNA301554.slim.GEM.log2.paf-th0.00-ple-3-rsq0.30-filtered" \
  --top_n 26035
```

Here, we provide the network filtered by Step 6 for the `--net` argument and the `--out_prefix` is used to name the resulting output file. The `--top_n` arguments allows us to use an *ad hoc* threshold to keep only the best ranked edges. This example network is small, so the `--top_n` argument is set to the total number of edges. By default, the `--top_n` argument is set to 10,000. You can use this to filter the best edges when networks become extremely large.

To create individual files for each condition add the `--save_condition_networks` argument. The resulting file will include the top n edges per condition not just the top n for the entire network:

```
kinc-filter-rank.R \
  --net "PRJNA301554.slim.GEM.log2.paf-th0.00-ple-3-rsq0.30-filtered.GCN.txt" \
  --out_prefix "PRJNA301554.slim.GEM.log2.paf-th0.00-ple-3-rsq0.30-filtered" \
  --save_condition_networks \
  --top_n 26035
```

If you are interested in exploring edges that are unique to a given category (e.g. heat or drought within a Treatment class) then you can provide the `--unique_filter` argument with the value "label":

```
kinc-filter-rank.R \
  --net "PRJNA301554.slim.GEM.log2.paf-th0.00-ple-3-rsq0.30-filtered.GCN.txt" \
  --out_prefix "PRJNA301554.slim.GEM.log2.paf-th0.00-ple-3-rsq0.30-filtered" \
  --save_condition_networks --unique_filter "label" \
  --top_n 26035
```

The result from the command-above is a set of files, one per condition class/label that contain only edges that are unique to the condition label (i.e. category) and is not significant for any other condition.

Finally, you can export the top n for a given condition class (e.g. Treatment) by providing the value "class" to the `--unique_filter` argument.

```
kinc-filter-rank.R \
  --net "PRJNA301554.slim.GEM.log2.paf-th0.00-ple-3-rsq0.30-filtered.GCN.txt" \
  --out_prefix "PRJNA301554.slim.GEM.log2.paf-th0.00-ple-3-rsq0.30-filtered" \
  --save_condition_networks --unique_filter "class" \
  --top_n 26035
```

The result from the command-above is a set of files, one per condition class where the top n edges per class are kept. An edge may be significant for multiple labels within the class but not for any other class.

7.3.9 Step 9: Visualization

You can visualize the network using 2D layout tools such as ‘[Cytoscape](https://cytoscape.org/)’, which is a feature rich 2D visualization software package. However, KINC includes a Python v3 Dash-based application that can be used for 3D visualization of the network. Please see the `_installation_reference_label` for the list of Python libraries that are required.

The following is an example for launching the viewer for the network containing all condition-specific subgraphs:

```
kinc-3d-viewer.py \  
  --net "PRJNA301554.slim.GEM.log2.paf-th0.00-ple-3-rsq0.30-filtered-th_ranked.  
↪csGCN.txt" \  
  --emx "PRJNA301554.slim.GEM.log2.txt" \  
  --amx "PRJNA301554.slim.annotations.txt"
```

Alternatively, you can view the condition-specific networks for the duration-specific subgraph:

```
kinc-3d-viewer.py \  
  --net "PRJNA301554.slim.GEM.log2.paf-th0.00-ple-3-rsq0.30-filtered-th_ranked.  
↪Duration-unique_class.csGCN.txt" \  
  --emx "PRJNA301554.slim.GEM.log2.txt" \  
  --amx "PRJNA301554.slim.annotations.txt"
```

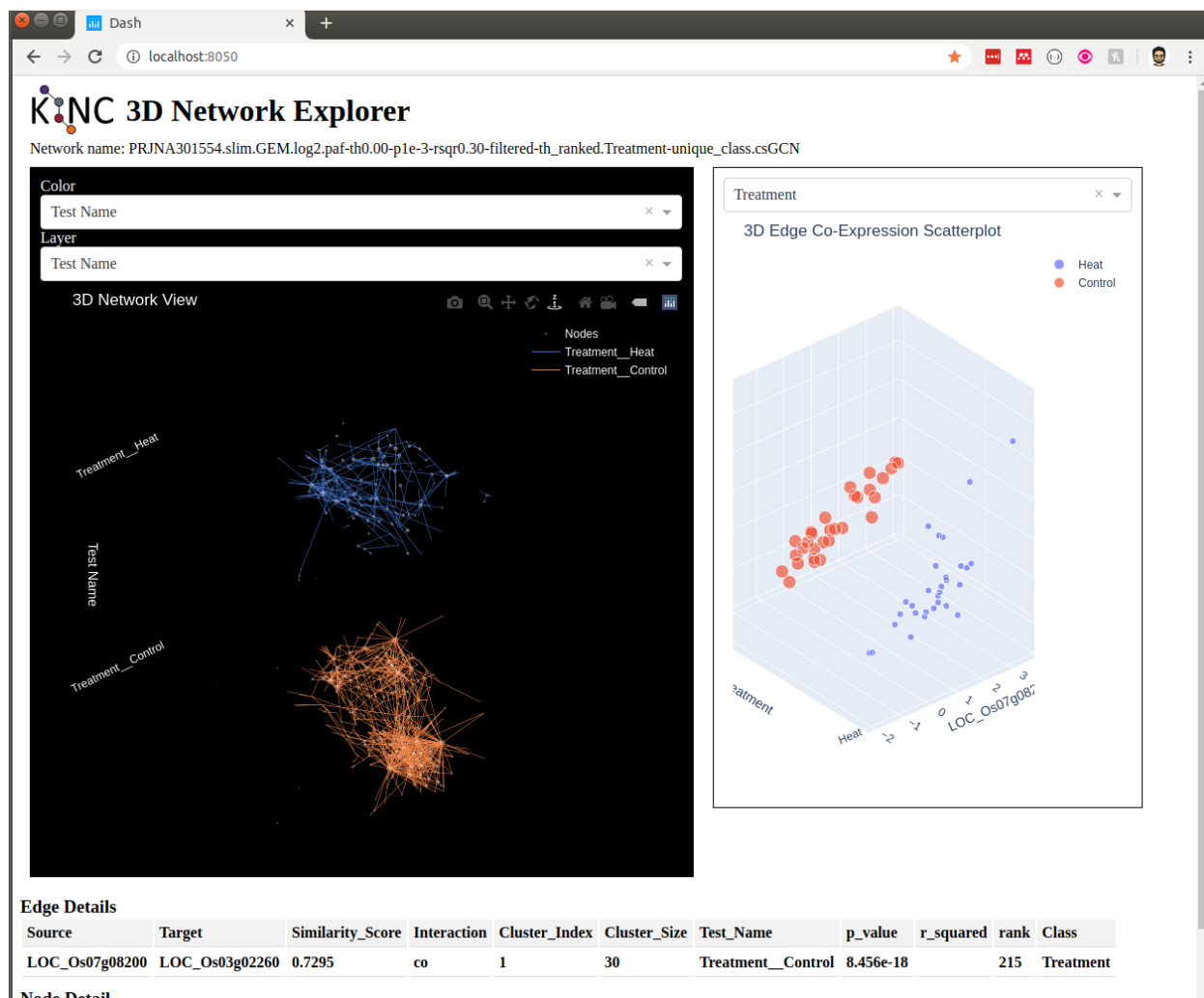
and for the treatment-specific subgraph:

```
kinc-3d-viewer.py \  
  --net "PRJNA301554.slim.GEM.log2.paf-th0.00-ple-3-rsq0.30-filtered-th_ranked.  
↪Treatment-unique_class.csGCN.txt" \  
  --emx "PRJNA301554.slim.GEM.log2.txt" \  
  --amx "PRJNA301554.slim.annotations.txt"
```

The first time the viewer is launched it will take a few moments to generate 2D and 3D layouts for the network. This will result in a set of new layout files created in the same folder as the network. These will only be generated once and will be re-used if the script is re-run. After creation of the layouts, a URL will be provided on the screen which should then be opened in a web browser. Output similar to the following should be seen in the terminal:

```
Reading network file...  
Reading GEM file...  
Reading experiment annotation file...  
Launching application...  
* Serving Flask app "kinc-3d-viewer" (lazy loading)  
* Environment: production  
  WARNING: This is a development server. Do not use it in a production deployment.  
  Use a production WSGI server instead.  
* Debug mode: off  
* Running on http://127.0.0.1:8050/ (Press CTRL+C to quit)
```

Finally, open the web browser to the specified URL to view the network.



Auxiliary Scripts

Included in the KINC repository are several Bash and Python scripts for a variety of helpful tasks. These scripts are documented here. For each of these scripts, you can run the script with no arguments (or with `-h` for Python scripts) to see the help text.

8.1 `kinc.sh`

The `kinc.sh` script can run each KINC analytic in a customizable sequence on any input GEM, using any runner (`serial`, `cuda`, `opencl`) and any number of processes via MPI. These three things are provided as command-line arguments. However, you can also set which analytics to run by setting the `DO_*` variables in the script source.

```
scripts/kinc.sh serial 1 GEM.txt
```

8.2 `kinc-py.sh`

The `kinc-py.sh` script does the same thing as `kinc.sh` but with Python scripts: `kinc-similarity.py`, `kinc-threshold.py`, and `kinc-extract.py`. These scripts can also be used individually.

```
scripts/kinc-py.sh GEM.txt
```

8.3 `make-input-data.py`

The `make-input-data.py` script creates a “fake” GEM given the number of genes, samples, and classes, as well as other customization options. This script is useful for testing KINC when you don’t have any real data on hand.

```
python scripts/make-input-data.py
```

8.4 validate.py

The `validate.py` script attempts to measure the difference between two similarity matrices by comparing the features of each edge (number of clusters, sample statistics, correlation value, sample mask, etc.). This script is useful for comparing similarity matrices from different runs to confirm that they are equivalent or to simply quantify their differences. You must provide the plain-text similarity matrix file, not the network file. This file can be generated using the `export-cmx` analytic.

```
python scripts/validate.py GEM1.cmx.txt GEM2.cmx.txt
```

8.5 visualize.py

The `visualize.py` script can create several useful visualizations of a network, such as scatter plots of the gene pairs in the network. Consult the help text to see all visualization options.

```
python scripts/visualize.py --emx GEM.txt --netlist GEM.coexpnet.txt --corrdist
```

There are issues that can occur when using KINC, ranging from data management to software and hardware configuration. The following sections will attempt to address the most common issues.

9.1 General Help and Guidance

If this online manual does not provide sufficient help or guidance, or if you encounter bugs or challenges please let us know. Please submit an issue on our **'KINC issue tracking'** <https://github.com/SystemsGenetics/KINC/issues> queue.

9.2 Thresholding Large GEMs

There are several issues that emerge when applying KINC to large input GEMs. Aside from the large computational cost in the similarity step, the main issue with large GEMs is the thresholding step. The `rmt` analytic finds a suitable threshold by performing a chi-squared test for each threshold to determine whether the thresholded network is random or non-random. In particular, the chi-squared value should transition from a low value (< 100) to a high value (> 200), and the threshold at which this transition occurs is selected as the final threshold. However, if the similarity matrix contains many edges even at high correlations (> 0.95), it will likely cause the chi-squared value to be too large for this transition to ever occur, and a suitable threshold will not be found.

There are many approaches to dealing with this issue. In general, the best thing to do first is to visualize some properties of the data: the sample distribution of the GEM (consult the `GEMprep` tool), the correlation distribution of the similarity matrix (using `visualize.py`), and pairwise scatter plots of the similarity matrix (using `visualize.py`). These visualizations can help you to identify irregularities or noise in the data, from which you can decide how to remove them. For example, if the GEM contains many negative-valued expressions that are noisy, you can use the `--minexpr` option in the `similarity` analytic to exclude these expressions when computing the pairwise correlations, which may remove many high but meaningless correlations.

CHAPTER 10

How to Cite KINC

If you find KINC useful and would like to cite it please consider the following:

The following paper introduces the GMM approach as implemented in KINC for condition-specific modeling:

- Sherman BT, Burns J, Feltus FA, Smith M, Ficklin SP. GPU Implementation of Pairwise Gaussian Mixture Models for Multi-Modal Gene Co-Expression Networks (2019). IEEE Access. 7, 160845-160857

A second manuscript describing the full KINC pipeline for a Bioinformatic audience is currently in preparation. If you prefer to refer directly to the software itself please consider the following:

- Josh Burns, Ben Shealy, Alex Feltus, Melissa Smith, & Stephen Ficklin. (2019, June 25). SystemsGenetics/KINC: Version 3.3.0 (Version v3.3.0). Zenodo. <http://doi.org/10.5281/zenodo.3256358>

Publications Using KINC

The following publications use KINC:

- McKnight CB, Poulos AL, Bender MR, Calhoun JC, Feltus FA. [Exploring Lossy Compression of Gene Expression Matrices](#) (2019). *IEEE/ACM 5th International Workshop on Data Analysis and Reduction for Big Scientific Data* (DRBSD-5). 28-34
- Shealey B, Burns JJR, Smith MC, Feltus FA, Ficklin SP. [GPU Implementation of Pairwise Gaussian Mixture Models for Multi-Modal Gene Co-Expression Networks](#) (2019). *IEEE Access*. 7:160845-160857.
- Poehlman WL, Schnabel E, Chavan S, Frugoli JA, Feltus FA. [Identifying Temporally Regulated Root Nodulation Biomarkers Using Time Series Gene Co-Expression Network Analysis](#) (2019). *Frontiers in Plant Science*. 10, p1409.
- Honaas LA, Hargarten HL, Ficklin SP, Hadish JA, Wafula E, Mattheis JP, Rudell DR. [Co-expression networks provide insights into molecular mechanisms of postharvest temperature modulation of apple fruit to reduce superficial scald](#) (2019). *Postharvest Biology and Technology*, 149:27-41.
- Dunwoodie LJ, Poehlman WL, Ficklin SP, Feltus FA. [Discovery and validation of a glioblastoma co-expressed gene module](#) (2018). *Oncotarget*. 9(13):10995–11008
- Ficklin SP, Dunwoodie LJ, Poehlman WL, Watson C, Roche KE, Feltus FA. [Discovering condition-specific gene co-expression patterns using gaussian mixture models: a cancer case study](#) (2017) *Scientific Reports*. 7: 8617
- Poehlman WL, Rynge M, Balamurugan D, Nills M, Feltus FA. [OSG-KINC: High-throughput gene co-expression network construction using the open science grid](#) (2017). *IEEE International Conference on Bioinformatics and Biomedicine* (BIBM), 1827-1831.