KINC v0.1 Specification

Knowledge Independent Network Construction

Joshua Burns*, Stephen Ficklin*

November 20, 2015

*Dept. of Horticulture, Washington State University

Contents

1 Introduction

KINC is designed for use in construction of biological networks, specifically, gene co-expression networks.

KINC performs three major steps:

1) Construction of a similarity matrix of pair-wise expression correlations.
2) Thresholding of the similarity matrix to form an adjaceny matrix.
3) Export of the adjaceny matrix to form a tab-delimited network file.

This document provides an overview for the data structures and file formats used by KINC.

Can we describe here how the command-line for KINC will work and how use of the classes described below translates into command-line arguments?

Yes I will add descriptions of command line arguments that will apply to all abstract classes.

2 Abstract Classes

The following abstract classes serve as the base for all structures within the kinc program. All other classes should inherit from these. This design will allow for dynamic addition of user implemented plugins in the future, because the functions of the abstract classes are clearly defined.

2.1 KINCData

I am thinking the header of a file, specifically its KINCData type, user defined name, and history, should all be defined here since any KINC data file will share all of these things in their file data.

The KINCData class is responsible for reading and writing KINC data files. Classes that inherit from the KINCData class are responsible for implementing the generic functions exposed by this class. This includes reading, writing, merging, exporing, indexing and querying. Typically, child classes provide importers that read commonly used file formats into their own binary file format and exporters to convert back to those same file formats.

This abstract class is responsible for reading in the header information of any data file since it is generic to any specific data type. Figure 1 shows the binary format for the beginning of any KINC data file. headerTag represents the specific data type this file represents. The rest of the header information is the history information for this data, starting with how this specific data was created.

Field	Description	Type
headerTag	Special header string that defines KINCData type for a file.	char[10]
history	Array of history items.	byte[]

Figure 1: Binary File Format of Header

Figure 2 shows the format for a single history item. The history item structure is a nested structure with subhistories of all input files used to create the current history item. This nesting continues until you reach the origional files that were created through importation of data outside of KINC. The first history item in any file is a reference of itself, therefore any data file has at least one history item which describes itself. Any sub history items after that describe any input files that helped create this file.

file is the filename of the input file or an empty string if this is the first history item referencing itself. name is the user defined name of the file and its data. object is teh name of the KINC Analytic or Data class which created this file. description is a detailed description about the creation of the data. Lastly, date is a linux timestamp of the time this file was created or last modified.

The subHistory items represent a list of further history items that describe all input files used in the creation of this current history item.

Field	Description	Type
fileLen	Length of file name string in bytes.	uint16_t
nameLen	Length of user defined name string in bytes.	uint16_t
objectLen	Length of object name string in bytes.	uint16_t
descriptionLen	Length of description string in bytes.	uint16_t
date	Linux time-stamp of when file was built.	uint64_t
file	File name string.	char[fileLen]
name	User defined name string.	char[nameLen]
object	Name of object that built file in bytes.	char[objectLen]
description	Detailed description of how file was built.	char[descriptionLen]
subHistorySize	Size of input history data in bytes.	uint32_t
subHistoryAmt	Number of input history items.	uint16_t
subHistory	Array of input history items.	byte[historySize]

Figure 2: Binary File Format of Individual History Item

2.2 KINCAnalytic

The KINCAnalytic class is responsible for taking in one or more KINCData objects and employ an algorithm such as a statistical test to produce one or more new KINCData objects. A KINCAnalytic

3 KINCData Classes

3.1 Expression

The Expression class is responsible for manging gene expression-level data.

3.1.1 Properties

Do we need any properties?

I don't think we need properties. I am also unsure how to implement them in C++. My thought is all interactions between the classes will be defined in the Abstract Classes section using virtual functions?

3.1.2 Constructor

KINCData(int argc, char *argv[])

We need to design how the functions of the class will receive arguments. will we have a constructor that receives, parses and responds to errors for all functions? Or should each function be responsible for checking it's own arguments. I know we can't do that in the abstract class, but we need to accomdate the behavior we settle on in our design so plugins are consistent.

I almost completely agree. These functions and interactions will all be defined in the abstract classes section if that is OK? It is standard C++ to define everything you are talking about in the abstract interface class with virtual functions. It is usually a good idea to have a default constructor only for implemenation classes, and have any additional configuation added into additional virtual functions that any implementation must handle.

3.1.3 Virtual Functions

The following functions should be implemented by any plugin that creates classes that inherits the KINCData class.

virtual void import() = 0

This function reads a tab-delimited file. Each line of this file represents the gene expression levels of a single gene, transcript or probeset. Each tab-separated value in a single line indicates the gene expression level for each sample. The expression level of a samples must be in the same order for every line. The first line of the file may contain a tab-delimited list of sample names, and a file may contain as many samples and genes as desired.

virtual void export() = 0

virtual void query() = 0

virtual void merge() = 0

3.1.4 File Structure

Figure 3 shows the binary format of expression data and how it is stored on file. geneAmt and sampleAmt give the total number of genes and samples in the data, respectively. geneNames is the list of all gene names as a string who's length and partitioning is defined by geneNameLen and geneNameSize. sampleNames is the list of all sample names as a string who's length and partitioning is defined by sampleNameLen and sampleNameSize. Lastly, samples is 2 dimensional matrix of all samples for each gene, where the matrix is sorted by gene major order.

Field	Description	Type
geneAmt	Total number of genes.	uint32_t
sampleAmt	Number of samples per gene.	uint32_t
geneNameLen	Length of each string identifying genes.	uint16_t
geneNameSize	Total size of gene name list in bytes.	uint64_t
sampleNameLen	Length of each string identifying samples.	uint16_t
sampleNameSize	Total size of sample name list in bytes.	uint64_t
geneNames	List of gene string identifiers.	char[geneNameSize]
sampleNames	List of sample string identifiers.	char[sampleNameSize]
sampleTotal	Total number of samples for all genes.	uint64_t
samples	List of all samples per gene.	float[sampleTotal]

Figure 3: Binary File Format of Expression Data

3.2 Correlation

This is responsible for storing correlation data between genes.

The following describes the format of the KINC correlation file. All multi-byte numbers are little-endian, regardless of the machine endianness.

I like this type of table for describing the file format. I borrowed it from the BAM file specification

So do I! I was actually going to convert these defintions to a tabular format after your first review. :)

Figure 4 shows the binary format of correlation data and how it is stored on file. geneAmt, sampleAmt, and corrAmt give the number of genes, number of samples per gene, and number of correlations per gene, respectively. geneNames is the list of all gene names that are correlated who's length and partitioning is defined by geneNameLen and geneNameSize. sampleNames is the list of all sample names used for correlation between genes who's length and partitioning is defiend by sampleNameLen and sampleNameSize. corrTypes is the list of all correlation types listed for all gene pairs who's length and partitioning is defined by corrTypeLen and corrTypeSize. Lastly, correlations is a special diagonal matrix where all correlations for gene pairs are stored using gene major order.

Field	Description	Type
geneAmt	Total number of genes.	uint32_t
sampleAmt	Number of samples per gene.	uint32_t
corrAmt	Number of correlations per gene relationship.	uint8_t
geneNameLen	Length of each string identifying genes.	uint16_t
geneNameSize	Total size of gene name list in bytes.	uint64_t
sampleNameLen	Length of each string identifying samples.	uint16_t
sampleNameSize	Total size of sample name list in bytes.	uint64_t
corrTypeLen	Length of each string identifying correlation type.	uint16_t
corrTypeSize	Total size of correlation type list in bytes.	uint16_t
geneNames	List of gene string identifiers.	char[geneNameSize]
sampleNames	List of sample string identifiers.	char[sampleNameSize]
corrTypes	List of correlation type strings.	char[corrTypeSize]
corrTotal	Total number of correlations for all gene relationships.	uint64_t
correlations	Diagonal matrix list of all gene correlations for all relationships.	float[corrTotal]

Figure 4: Binary File Format of Correlation Data

3.3 Network

This is responsible for storing network data between genes.

Figure 5 shows the binary format of network data and how it is stored on file. geneAmt give the number of genes in the network. geneNames is the list of all gene names that are correlated who's length and partitioning is defined by geneNameLen and geneNameSize. Lastly, network is a special diagonal matrix where all network edges for gene pairs are stored using gene major order.

Field	Description	Type
geneAmt	Total number of genes.	uint32_t
geneNameLen	Length of each string identifying genes.	uint16_t
geneNameSize	Total size of gene name list in bytes.	uint64_t
geneNames	List of gene string identifiers.	char[geneNameSize]
netTotal	Total number of edges, true or false, in network data.	uint64_t
network	Diagonal matrix list of all possible edges in gene network.	bool[netTotal]

Figure 5: Binary File Format of Network Data

3.4 Annotation

This is responsible for storing additional information for genes.

Figure 6 shows the binary format of annotation data and how it is stored on file. geneAmt and annotAmt give the number of genes and the number of annotations, respectively. geneNames is the list of all gene names who's length and partitioning is defined by geneNameLen and geneNameSize.annotNames is the list of all annotation names who's length and partitioning is defined by annotNameLen and annotNameSize. annotValSize is a list of all string lengths for each annotation value per each gene. Lastly, annotations is a 2 dimensional matrix that lists all annotations for all genes using gene major order.

Field	Description	Type
geneAmt	Total number of genes.	uint32_t
annotAmt	Total number of annotations per gene.	uint32_t
geneNameLen	Length of each string identifying genes.	uint16_t
geneNameSize	Total size of gene name list in bytes.	uint64_t
annotNameLen	Length of each string identifying the name of a annotation.	uint16_t
annotNameSize	Total size of annotation name list in bytes.	uint64_t
annotNames	List of annotation string identifiers.	char[annotNameSize]
geneNames	List of gene string identifiers.	char[geneNameSize]
annotValLens	List of numbers that identify the length of each value string for each annotation.	uint16_t[annotAmt]
annotValSize	Total size of all annotation values.	uint64_t
annotations	List of all annotations per gene.	char[annotValSize]

Figure 6: Binary File Format of Annotation Data

4 KINCAnalytic Classes

4.1 Pearson

This takes an Expression BioData object and produces a Correlation BioData object. It uses the Pearson correlation statistical method for giving correlation values.

4.2 Spearman

This takes an Expression BioData object and produces a Correlation BioData object. It uses the Spearman correlation statistical method for giving correlation values.