PCM

August 6, 2015

Title PCM: A pairwise correlation mining package for biological network inference.
Version 0.1.0
Author Zengyou He , Feiyang Gu and Xiaoqing Liu
Language C++(include Eigen library)
Description Procedures to pairwise correlation mining using various types of correlation measures. Source code is available at https://github.com/FeiyangGu/PCM.
Maintainer Zengyou He <zyhe@dlut.edu.cn> Feiyang Gu <gufeiyang1000@gmail.com>

The format of input Columns and rows in the input table correspond to variables (e.g. genes, proteins) and samples (e.g. gene expression profiles), respectively. The data in the same row are separated by a space. The values of binary data should be 0 or 1 and the values of disperse data should be transformed integers, which begin with 0. For continuous data, there is no special demand.

The format of output Every row contains a pair of variables whose coefficient stratifies the constraint of demand and the corresponding value of coefficient. But for LOPCC, there is no specific value of coefficient.

Sample data We provide two sample data. One is a network inference data, DREAM3, which has 100 rows and 100 columns. DREAM3 is a continuous data and is named as data_DREAM3.txt in data file. The other is also a PPI network inference data, Gavin et al. We transform this data to a binary table and name it as data_Gavin.txt in the data file. It contains 2166 rows and 2761 columns.

Functions

void SupportB(char* inputfile, char *outputfile, double threshold). SupportB is the implementation of Support for binary data. "inputfile" and "outputfile" are the file paths of input data and output data, respectively. SupportB can find all pairs whose Support's correlation coefficients are no less than threshold. The usage of SupportB is:

```
PCM.exe SupportB input.txt output.txt 0.2
```

void JaccardB (char* inputfile, char *outputfile, double threshold). JaccardB is the implementation of Jaccard for binary data. "inputfile" and "outputfile" are the file paths of input data and output data, respectively. JaccardB can find all pairs whose Jaccard's correlation coefficients are no less than threshold. The usage of JaccardB is:

```
PCM.exe JaccardB input.txt output.txt 0.2
```

void InterestB (char* inputfile, char *outputfile, double threshold). InterestB is the implementation of Interest for binary data. "inputfile" and "outputfile" are the file paths of input data and output data, respectively. InterestB can find all pairs whose Interest's correlation coefficients are no less than threshold. The usage of InterestB is:

```
PCM.exe InterestB input.txt output.txt 0.2
```

void Piatetsky_ShapirosB (char* inputfile, char *outputfile, double threshold). Piatetsky_ShapirosB is the implementation of Piatetsky_Shapiro's for binary data. "inputfile" and "outputfile" are the file paths of

input data and output data, respectively. Piatetsky_ShapirosB can find all pairs whose Piatetsky_Shapiro's interset's correlation coefficients are no less than threshold. The usage of Piatetsky_ShapirosB is:

PCM.exe Piatetsky_ShapirosB input.txt output.txt 0.2

void CosineB (char* inputfile, char *outputfile, double threshold). CosineB is the implementation of Cosine for binary data. "inputfile" and "outputfile" are the file paths of input data and output data, respectively. CosineB can find all pairs whose Cosine's correlation coefficients are no less than threshold. The usage of CosineB is:

PCM.exe CosineB input.txt output.txt 0.2

void ConfidenceB (char* inputfile, char *outputfile, double threshold). ConfidenceB is the implementation of Confidence for binary data. "inputfile" and "outputfile" are the file paths of input data and output data, respectively. ConfidenceB can find all pairs whose Confidence's correlation coefficients are no less than threshold. The usage of ConfidenceB is:

PCM.exe ConfidenceB input.txt output.txt 0.2

void YulesQB (char* inputfile, char *outputfile, double threshold). YulesQB is the implementation of Yule's Q for binary data. "inputfile" and "outputfile" are the file paths of input data and output data, respectively. YulesQB can find all pairs whose Yule's Q's correlation coefficients are no less than threshold. The usage of YulesQB is:

PCM.exe YulesQB input.txt output.txt 0.2

void YulesYB (char* inputfile, char *outputfile, double threshold). YulesYB is the implementation of Yule's Y for binary data. "inputfile" and "outputfile" are the file paths of input data and output data, respectively. YulesYB can find all pairs whose Yule's Y's correlation coefficients are no less than threshold. The usage of YulesYB is:

PCM.exe YulesYB input.txt output.txt 0.2

void KappaB (char* inputfile, char *outputfile, double threshold). KappaB is the implementation of Kappa for binary data. "inputfile" and "outputfile" are the file paths of input data and output data, respectively. KappaB can find all pairs whose Kappa's correlation coefficients are no less than threshold. The usage of KappaB is:

PCM.exe KappaB input.txt output.txt 0.2

void J_MeasureB (char* inputfile, char *outputfile, double threshold). J_MeasureB is the implementation of J-measure for binary data. "inputfile" and "outputfile" are the file paths of input data and output data, respectively. J_MeasureB can find all pairs whose J-measure's correlation coefficients are no less than threshold. The usage of J_MeasureB is:

PCM.exe J_MeasureB input.txt output.txt 0.2

void OddsRatioB (char* inputfile, char *outputfile, double threshold). OddsRatioB is the implementation of Odds ratio for binary data. "inputfile" and "outputfile" are the file paths of input data and output data, respectively. OddsRatioB can find all pairs whose Odds ratio's correlation coefficients are no less than threshold. The usage of OddsRatioB is:

PCM.exe OddsRatioB input.txt output.txt 0.2

void GiniIndexB (char* inputfile, char *outputfile, double threshold). GiniIndexB is the implementation of Gini index for binary data. "inputfile" and "outputfile" are the file paths of input data and output data, respectively. GiniIndexB can find all pairs whose Gini index's correlation coefficients are no less than threshold. The usage of GiniIndexB is:

PCM.exe GiniIndexB input.txt output.txt 0.2

void LaplaceB (char* inputfile, char *outputfile, double threshold). LaplaceB is the implementation of

Laplace for binary data. "inputfile" and "outputfile" are the file paths of input data and output data, respectively. LaplaceB can find all pairs whose Laplace's correlation coefficients are no less than threshold. The usage of LaplaceB is:

PCM.exe LaplaceB input.txt output.txt 0.2

void ConvictionB (char* inputfile, char *outputfile, double threshold). ConvictionB is the implementation of Conviction for binary data. "inputfile" and "outputfile" are the file paths of input data and output data, respectively. ConvictionB can find all pairs whose Conviction's correlation coefficients are no less than threshold. The usage of ConvictionB is:

PCM.exe ConvictionB input.txt output.txt 0.2

void CertaintyFactorB (char* inputfile, char *outputfile, double threshold). CertaintyFactorB is the implementation of Certainty factor for binary data. "inputfile" and "outputfile" are the file paths of input data and output data, respectively. CertaintyFactorB can find all pairs whose Certainty factor's correlation coefficients are no less than threshold. The usage of CertaintyFactorB is:

PCM.exe CertaintyFactorB input.txt output.txt 0.2

void AddedValueB (char* inputfile, char *outputfile, double threshold). AddedValueB is the implementation of Added value for binary data. "inputfile" and "outputfile" are the file paths of input data and output data, respectively. AddedValueB can find all pairs whose Added value's correlation coefficients are no less than threshold. The usage of AddedValueB is:

PCM.exe AddedValueB input.txt output.txt 0.2

void KlosgenB (char* inputfile, char *outputfile, double threshold). KlosgenB is the implementation of Klosgen for binary data. "inputfile" and "outputfile" are the file paths of input data and output data, respectively. KlosgenB can find all pairs whose Klosgen's correlation coefficients are no less than threshold. The usage of KlosgenB is:

PCM.exe KlosgenB input.txt output.txt 0.2

void Phi_CoefficientB (char* inputfile, char *outputfile, double threshold). Phi_CoefficientB is the implementation of ∅- coefficient for binary data. "inputfile" and "outputfile" are the file paths of input data and output data, respectively. Phi_CoefficientB can find all pairs whose ∅- coefficient's correlation coefficients are no less than threshold. The usage of Phi_CoefficientB is:

PCM.exe Phi_CoefficientB input.txt output.txt 0.2

void ProbabilityRatioB (char* inputfile, char *outputfile, double threshold). ProbabilityRatioB is the implementation of Probability Ratio for binary data. "inputfile" and "outputfile" are the file paths of input data and output data, respectively. ProbabilityRatioB can find all pairs whose Probability Ratio's correlation coefficients are no less than threshold. The usage of ProbabilityRatioB is:

PCM.exe ProbabilityRatioB input.txt output.txt 0.2

void LikelihoodRatioB (char* inputfile, char *outputfile, double threshold). LikelihoodRatioB is the implementation of Likelihood Ratio for binary data. "inputfile" and "outputfile" are the file paths of input data and output data, respectively. LikelihoodRatioB can find all pairs whose Likelihood Ratio's correlation coefficients are no less than threshold. The usage of LikelihoodRatioB is:

PCM.exe LikelihoodRatioB input.txt output.txt 0.2

void BCPNNB(char* inputfile, char *outputfile, double threshold, double cc). BCPNNB is the implementation of BCPNN for binary data. "inputfile" and "outputfile" are the file paths of input data and output data, respectively. The default value of cc is 0.5/n (n is the number of rows). BCPNNB can find all pairs whose BCPNN's correlation coefficients are no less than threshold. The usage of BCPNNB is:

PCM.exe BCPNNB input.txt output.txt 0.2

or PCM.exe SCWCCB input.txt output.txt 0.2 0.01

void SCWCCB (char* inputfile, char *outputfile, double threshold, double cc). SCWCCB is the implementation of SCWCC for binary data. "inputfile" and "outputfile" are the file paths of input data and output data, respectively. The default value of cc is 0.5/n (n is the number of rows). SCWCCB can find all pairs whose SCWCC's correlation coefficients are no less than threshold. The usage of SCWCCB is:

PCM.exe SCWCCB input.txt output.txt 0.2

or PCM.exe SCWCCB input.txt output.txt 0.2 0.01

void TwoWaySupportB (char* inputfile, char *outputfile, double threshold). TwoWaySupportB is the implementation of Two-way Support for binary data. "inputfile" and "outputfile" are the file paths of input data and output data, respectively. TwoWaySupportB can find all pairs whose Two-way Support's correlation coefficients are no less than threshold. The usage of TwoWaySupportB is:

PCM.exe TwoWaySupportB input.txt output.txt 0.2

void SCWSB (char* inputfile, char *outputfile, double threshold). SCWSB is the implementation of SCWS for binary data. "inputfile" and "outputfile" are the file paths of input data and output data, respectively. SCWSB can find all pairs whose SCWS's correlation coefficients are no less than threshold. The usage of SCWSB is:

PCM.exe SCWSB input.txt output.txt 0.2

void SimplifiedXstatisticB (char* inputfile, char *outputfile, double threshold). SimplifiedXstatisticB is the implementation of Simplified χ 2-statistic for binary data. "inputfile" and "outputfile" are the file paths of input data and output data, respectively. SimplifiedXstatisticB can find all pairs whose Simplified χ 2-statistic's correlation coefficients are no less than threshold. The usage of SimplifiedXstatisticB is:

PCM.exe SimplifiedXstatisticB input.txt output.txt 0.2

void CMI2NIB(char* inputfile, char *outputfile, double threshold, int order0 = 1). CMI2NIB is the implementation of CMI2NI for binary data. "inputfile" and "outputfile" are the file paths of input data and output data, respectively. The default value of order0 is 1. The usage of CMI2NIB is:

PCM.exe CMI2NIB input.txt output.txt 0.02

or PCM.exe CMI2NIB input.txt output.txt 0.02 2

void MID (char* inputfile, char *outputfile, double threshold). MID is the implementation of mutual information for discrete data. "inputfile" and "outputfile" are the file paths of input data and output data, respectively. MID can find all pairs whose mutual information's correlation coefficients are no less than threshold. The usage of MID is:

PCM.exe MID input.txt output.txt 0.2

void DiceC (char* inputfile, char *outputfile, double threshold). DiceC is the implementation of Dice for continuous data. "inputfile" and "outputfile" are the file paths of input data and output data, respectively. DiceC can find all pairs whose Dice's correlation coefficients are no less than threshold. The usage of DiceC is:

PCM.exe DiceC input.txt output.txt 0.2

void OverlapC (char* inputfile, char *outputfile, double threshold). OverlapC is the implementation of Overlap for continuous data. "inputfile" and "outputfile" are the file paths of input data and output data, respectively. OverlapC can find all pairs whose Overlap's correlation coefficients are no less than threshold. The usage of OverlapC is:

PCM.exe OverlapC input.txt output.txt 0.2

void CosineC (char* inputfile, char *outputfile, double threshold). CosineC is the implementation of Cosine for continuous data. "inputfile" and "outputfile" are the file paths of input data and output data, respectively. CosineC can find all pairs whose Cosine's correlation coefficients are no less than threshold. The usage of CosineC is:

PCM.exe CosineC input.txt output.txt 0.2

void JaccardC (char* inputfile, char *outputfile, double threshold). JaccardC is the implementation of Jaccard for continuous data. "inputfile" and "outputfile" are the file paths of input data and output data, respectively. JaccardC can find all pairs whose Jaccard's correlation coefficients are no less than threshold. The usage of JaccardC is:

PCM.exe JaccardC input.txt output.txt 0.2

void PearsonC (char* inputfile, char *outputfile, double threshold). PearsonC is the implementation of Pearson for continuous data. "inputfile" and "outputfile" are the file paths of input data and output data, respectively. PearsonC can find all pairs whose Pearson's correlation coefficients are no less than threshold. The usage of PearsonC is:

PCM.exe PearsonC input.txt output.txt 0.2

void SpearmanC(char* inputfile, char *outputfile, double threshold). SpearmanC is the implementation of Spearman for continuous data. "inputfile" and "outputfile" are the file paths of input data and output data, respectively. SpearmanC can find all pairs whose Spearman's correlation coefficients are no less than threshold. The usage of SpearmanC is:

PCM.exe SpearmanC input.txt output.txt 0.2

void DotProductC(char* inputfile, char *outputfile, double threshold). DotProductC is the implementation of Dot Product for continuous data. "inputfile" and "outputfile" are the file paths of input data and output data, respectively. DotProductC can find all pairs whose Dot Product's correlation coefficients are no less than threshold. The usage of DotProductC is:

PCM.exe DotProductC input.txt output.txt 0.2

void KendallC (char* inputfile, char *outputfile, double threshold). KendallC is the implementation of Kendall for continuous data. "inputfile" and "outputfile" are the file paths of input data and output data, respectively. KendallC can find all pairs whose Kendall's correlation coefficients are no less than threshold. The usage of KendallC is:

PCM.exe KendallC input.txt output.txt 0.2

void HoeffdingDC (char* inputfile, char *outputfile, double threshold). HoeffdingDC is the implementation of Hoeffding's D measure for continuous data. "inputfile" and "outputfile" are the file paths of input data and output data, respectively. HoeffdingDC can find all pairs whose Hoeffding's D measure's correlation coefficients are no less than threshold. The usage of HoeffdingDC is:

PCM.exe HoeffdingDC input.txt output.txt 0.2

void CMI2NIC(char* inputfile, char *outputfile, double threshold, int order0 = 2). CMI2NIC is the implementation of CMI2NI for continuous data. "inputfile" and "outputfile" are the file paths of input data and output data, respectively. The default value of order0 is 1. The usage of CMI2NIC is:

PCM.exe CMI2NIB input.txt output.txt 0.02

or PCM.exe CMI2NIB input.txt output.txt 0.02 2

void LOPCC(char* inputfile, char *outputfile). LOPCC is the implementation of LOPC for continuous data. "inputfile" and "outputfile" are the file paths of input data and output data, respectively. The usage of LOPCC is:

PCM.exe LOPCC input.txt output.txt

We also provide some other interfaces which can calculate the correlation between two variables for the users who want to utilize the source code.

The names of interfaces which begin with “BK” is for binary data and the names which beigin with “CK” is for continuous data.

The interfaces are:

```
double BK_Support(vector<int>& var1, vector<int>& var2);
double BK_Jaccard(vector<int>& var1, vector<int>& var2);
double BK_Interest(vector<int>& var1, vector<int>& var2);
double BK_Piatetsky_Shapiros(vector<int>& var1, vector<int>& var2);
double BK_Cosine(vector<int>& var1, vector<int>& var2);
double BK_Confidence(vector<int>& var1, vector<int>& var2);
double BK_YulesQ(vector<int>& var1, vector<int>& var2);
double BK_YulesY(vector<int>& var1, vector<int>& var2);
double BK_Kappa(vector<int>& var1, vector<int>& var2);
double BK_J_Measure(vector<int>& var1, vector<int>& var2);
double BK_OddsRatio(vector<int>& var1, vector<int>& var2);
double BK_GiniIndex(vector<int>& var1, vector<int>& var2);
double BK_Laplace(vector<int>& var1, vector<int>& var2);
double BK_Conviction(vector<int>& var1, vector<int>& var2);
double BK_CertaintyFactor(vector<int>& var1, vector<int>& var2);
double BK_AddedValue(vector<int>& var1, vector<int>& var2);
double BK_CollectiveStrength(vector<int>& var1, vector<int>& var2);
double BK_Klosgen(vector<int>& var1, vector<int>& var2);
double BK_Phi_Coefficient(vector<int>& var1, vector<int>& var2);
double BK_ProbabilityRatio(vector<int>& var1, vector<int>& var2);
double BK_LikelihoodRatio(vector<int>& var1, vector<int>& var2);
double BK_BCPNN(vector<int>& var1, vector<int>& var2, double cc);
double BK_SCWCC(vector<int>& var1, vector<int>& var2, double cc);
double BK_TwoWaySupport(vector<int>& var1, vector<int>& var2);
double BK_SCWS(vector<int>& var1, vector<int>& var2);
double BK_SimplifiedXstatistic(vector<int>& var1, vector<int>& var2);

double CK_Dice(vector<double>& var1, vector<double>& var2);
double CK_Jaccard(vector<double>& var1, vector<double>& var2);
double CK_Overlap(vector<double>& var1, vector<double>& var2);
double CK_Cosin(vector<double>& var1, vector<double>& var2);
double CK_Pearson(vector<double>& var1, vector<double>& var2);
double CK_Spearman(vector<double>& var1, vector<double>& var2);
double CK_DotProduct(vector<double>& var1, vector<double>& var2);
double CK_Kendall(vector<double>& var1, vector<double>& var2);
double CK_Distance(vector<double>& var1, vector<double>& var2);
double CK_HoeffdingD(vector<double>& var1, vector<double>& var2);
```