A bioinformatic implementation of mutual information as a distance measure for identification of clusters of variables

C. Pardy[1] S. R. Wilson[2]

(Received 31 January 2011; revised 12 August 2011)

Abstract

The size of data sets produced in genetic experiments is steadily increasing. Very often there are many more variables than observations, leading to the so-called "large p, small n" problem. For such data, clustering and distance based procedures are useful tools for identifying groups of variables associated with outcomes of interest. We develop a novel approach using mutual information as a measure of distance (here dependency) between probability distributions that is valid for comparisons between pairs of variables that are both continuous, both discrete, or one of each. This gives an overall information matrix to be used as a distance matrix in clustering procedures and to define a so-called weighted network of associations between variables. We present computational aspects of implementing our procedures in R.

Contents

1 Introduction

Our approach extends that of Zhang and Horvath [10, 4, 5]. The objective here is to use network based concepts to describe associations within genetic datasets, with the goal of identifying groups of genes (which they term *modules*) that contain many genes with an above average level of association with important clinical outcomes. This is done by defining a *weighted network* where each gene is conceptualised as a *node* with the strength of connections between nodes given by a transformed correlation score (which we replace with mutual information). We develop our techniques using Zhang and Horvath's publicly available F2 intercross dataset containing single nucleotide polymorphisms (SNPs) and gene expression levels in female mouse liver tissue. We use the same dataset analysed by Ghazalpour et al. [5] containing 135 mice, 3421 genes and 1065 SNPS. The gene expression levels take continuous values, whereas the SNPs represent a mutation that has three categorical levels labelled A, H and B. A and B represent homozygous genotypes, AA and BB respectively, where A and B are two SNP alleles, H is the heterozygote AB.

Zhang and Horvath's network and modules were constructed using only gene expression data. We extend this by directly including SNPs in the network of associations. Mutual Information (MI), an information theoretic quantity measuring the dependency relationship between two probability distributions [1], is proposed as a measure of association for both continuous and categorical data. Elements of this approach were suggested by Dawys et al. [2]. MI has the additional advantage of invariance under non-linear transformations such as rescaling, whereas Pearson and Spearman correlations only measure linear and monotonic relationships respectively. We show that it is possible to define a valid MI measure between discrete and continuous variables and give some advice regarding a tractable implementation in the `R` statistical language, serving as an example for improving `R` computational performance in general. We aim to develop computational procedures fast enough for the future application of resampling procedures (such as the bootstrap) to deal with statistical aspects of our analyses.

1.1 Information measures

Let X is a discrete random variable with $\Pr(X = x) = p(x)$, the entropy of X is

$$H(X) = -\sum_x p(x)\log(p(x)) = -E_X[\log(p(x))].$$

For two discrete random variables X and Y with joint probability mass function $\Pr(X = x, Y = y) = p(x, y)$ the mutual information is

$$I(X;Y) = \sum_{x,y} p(x,y)\log\left(\frac{p(x,y)}{p(x)p(y)}\right) = E_{(X,Y)}\left[\log\left(\frac{p(x,y)}{p(x)p(y)}\right)\right] \tag{1}$$

For continuous variables we replace sums with integrals. For example [1], bivariate random normal variables with mean zero, common variance and correlation ρ have a MI of $-\frac{1}{2}\log(1-\rho^2)$.

2 A mixture model for the relationship between a SNP and a gene expression level

We model the distribution of the continuous variable as a mixture of conditional distributions for each level of the categorical variable.

Let X be a discrete random variable corresponding to the three possible levels of a SNP. Write $\Pr(X = x_i) = p_i$ for $i \in \{1, 2, 3\}$. We prefer to think of X as a set valued [6] random variable $\Omega_X \to \chi := \{x_1, x_2, x_3\} \equiv \{A, H, B\}$. Let Y represent the continuously measured gene expression level with density and distribution functions $f_Y(y)$ and $F_Y(y)$ (that is, $Y : \Omega_Y \to \mathbb{R}$). The joint distribution of X and Y is $F_{(X,Y)}(x, y)$ with corresponding density $f_{(X,Y)}(x, y)$. For each x_i we have a conditional distribution $Y \mid X = x_i$ with continuous density function $f_{Y|X=x_i}(y \mid x_i)$. We assume all integrals exist. This gives rise to the unusual probability space characterised by the joint distribution $F(X,Y) : (\Omega_X \times \Omega_Y) \to (\chi \times \mathbb{R})$, which is best thought of as three separate continuous (conditional) distributions.

We suppose that the density of Y is the mixture

$$f(y) = \sum_{i=1}^{3} p_i f(y \mid x_i).$$

By writing $f_X(x) = \sum_{i=1}^{3} p_i \delta_{x_i}(x)$ and $f(y \mid x) = \sum_{i=1}^{3} f(y \mid x_i)\delta_{x_i}(x)$ we express the joint density as

$$f(x,y) = \sum_{i=1}^{3} p_i f(y \mid x_i)\delta_{x_i}(x),$$

where $\delta_{x_i}(x) = \mathbf{1}_{\{x=x_i\}}$. It is straightforward to show Proposition 1 (details to be provided elsewhere).

Proposition 1. *The mutual information under this mixture model is*

$$I(X;Y) = \sum_{i=1}^{3} p_i \int_{y\in\mathbb{R}} f(y \mid x_i) \log\left(\frac{f(y \mid x_i)}{f(y)}\right) dy\,. \tag{2}$$

This result can be interpreted as

$$I(X;Y) = D(f(x,y) \parallel f(x)f(y)) = \sum_{i=1}^{3} p_i D(f(y \mid x_i) \parallel f(y)),$$

where

$$D(f_X \parallel f_Y) = \int_{\Omega} \log\left(\frac{f_X(u)}{f_Y(u)}\right) dF_X(u) = \int_{-\infty}^{\infty} f_X(u) \log\left(\frac{f_X(u)}{f_Y(u)}\right) du$$

is the Kullback–Leibler divergence from the distribution of random variable X to the distribution of Y. A similar result was shown by Dawys et al. [2] although not for our specific mixture model. Our computational approach solves issues [2] regarding the practical application of these procedures (that is, our approach can be automated and numerical integration is straightforward and fast using Simpson's rule).

3 Nonparametric estimation

Kernel smoothing is a nonparametric approach to the estimation of probability distributions [9] that requires the choice of a *smoothing parameter* (also called the *bandwidth*). We use kernel approaches as a basis for calculating MI scores, and automate the procedure by using a data driven 'Direct plug-in' estimator to find an optimal bandwidth as proposed by Sheather and Jones [8]. Computational approximations that significantly improve performance were given by Wand and Jones [9].

3.1 Our first estimator

Our first estimate for (2) uses a non-parametric Gaussian kernel smoother based on one used by Qiu et al. [7]. For a sample $\mathbf{z} = z_1, \ldots, z_j$, where $|\mathbf{z}| = M_z$ is the number of observations of $\mathbf{z}$, one version of the normal kernel density estimator is

$$\hat{f}(z) = \frac{1}{M_z} \sum_{j=1}^{M_z} \frac{1}{\sqrt{2\pi h^2}} \exp\left[-\frac{1}{2h^2}(z - z_j)^2\right], \tag{3}$$

where h is a smoothing parameter. For continuous samples $\mathbf{x}$ and $\mathbf{y}$, the estimator used in [7] is

$$\hat{I}(X;Y) = \frac{1}{M} \sum_i \log \frac{M \sum_j \exp\left[-\frac{1}{2h^2}\left((x_i - x_j)^2 + (y_i - y_j)^2\right)\right]}{\sum_j \exp\left[-\frac{1}{2h^2}(x_i - x_j)^2\right] \sum_j \exp\left[-\frac{1}{2h^2}(y_i - y_j)^2\right]} . \tag{4}$$

Using a similar approach, we estimate each element of the sum (2) by taking an average of sample kernel estimates

$$\hat{E}_{Y|X=x_i}\left[\log\left(\frac{f(y \mid x_i)}{f(y)}\right)\right] = \frac{1}{M_{x_i}}\sum_{k=1}^{M_{x_i}} \log \frac{\frac{1}{M_{x_i}}\sum_{j|X=x_i}\exp\left[-\frac{1}{2h^2}(y_k - y_j)^2\right]}{\frac{1}{M}\sum_j \exp\left[-\frac{1}{2h^2}(y_k - y_j)^2\right]}\,,$$

since, by the law of large numbers (LLN), $\hat{E}_{Y|X=x_i} \to E_{Y|X=x_i}$ as $n \to \infty$. This gives our first estimator

$$\hat{I}(X;Y) = \sum_i p(x_i)\frac{1}{M_{x_i}}\sum_{k=1}^{M_{x_i}} \log \frac{M\sum_{j|X=x_i}\exp\left[-\frac{1}{2h^2}(y_k - y_j)^2\right]}{M_{x_i}\sum_j \exp\left[-\frac{1}{2h^2}(y_k - y_j)^2\right]}\,. \qquad (5)$$

The LLN is required here as this estimator only takes values at the observed data points, and is in this sense a form of discretisation.

3.2 Full density estimation

As an alternative to (5) we estimate the density along a grid of points using the well known kernel density estimation methods described by Wand and Jones [9] and available in the R package KernSmooth.

3.2.1 Binned Kernel Density Estimate

The Binned Kernel Density Estimate (BKDE) uses a Fast Fourier Transform (FFT) based computational approximation that provides us with additional advantages [9, Appendix D]. By quantising the data to a regularly spaced grid of points, convolution operations required to calculate $\hat{h}$ are computed quickly by multiplication in frequency space. The inverse FFT then returns the conditional density estimates $\hat{f}(y \mid x_i)$ on a sequence of regularly spaced points which are quickly integrated using Simpson's rule. The use of normal kernels ensures the target density functions are sufficiently smooth. We simply use this to estimate the density functions $\hat{f}(y \mid x_i)$ and plug them into (2), estimating the p_i with observed relative frequencies.

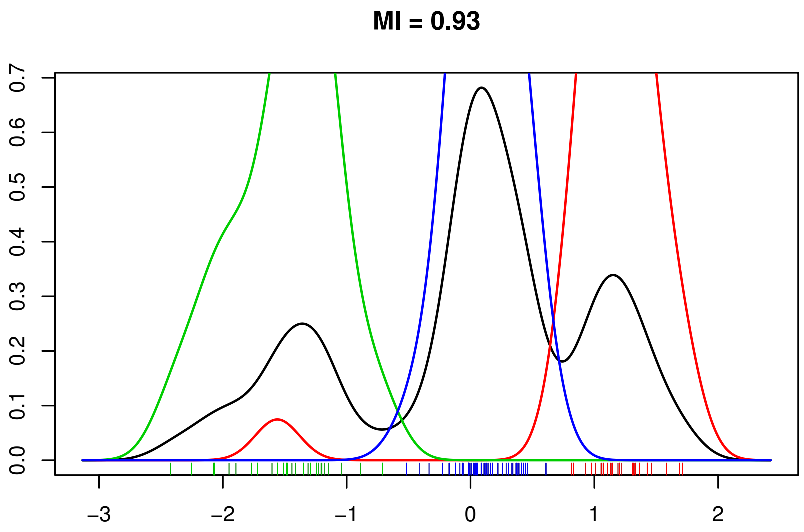
4 Interpretation of our mutual information estimators

Our estimators measure the degree of association between a continuous variable and a grouping factor. We aim for a high MI score when a continuous variable is clearly separated into groups by the categorical variable, and for low scores when there is no such relationship. An example from our dataset is shown in Figure 1. Here we use our second 'Full Density' estimation approach. Figure 1(a) shows three estimated conditional distributions with little overlap corresponding to a MI score of 0.93. Figure 1(b) shows the result of a random permutation of the group labels with substantial overlap of continuous distributions with the low MI score of 0.03.

5 A brief comparison of our two mutual information estimators

Two simulated cases are used to compare our estimators. For the first case, the levels of the SNP easily separate the conditional distributions of the gene expression level, giving a high MI score. Here the continuous variable is a mixture of three normal distributions with unit variance and means -5, 0 and 1. The second case is designed to show considerable overlap, using gamma distributions with means 5, 6 and 7, with variances all equal to 5. Exact MI values are calculated by numerically integrating the appropriate forms of (2) using the `R` function `integrate()`. Figure 2 shows boxplots of estimates over 1000 simulations, with horizontal lines showing the target values of 1.047 and 0.081 respectively. The LLN based approach (5) is more accurate when the groups are easily separated; however, it overestimates weak associations. Due to the considerable noise in genetic datasets, we prefer the more conservative full density approach.

(a)



(b)

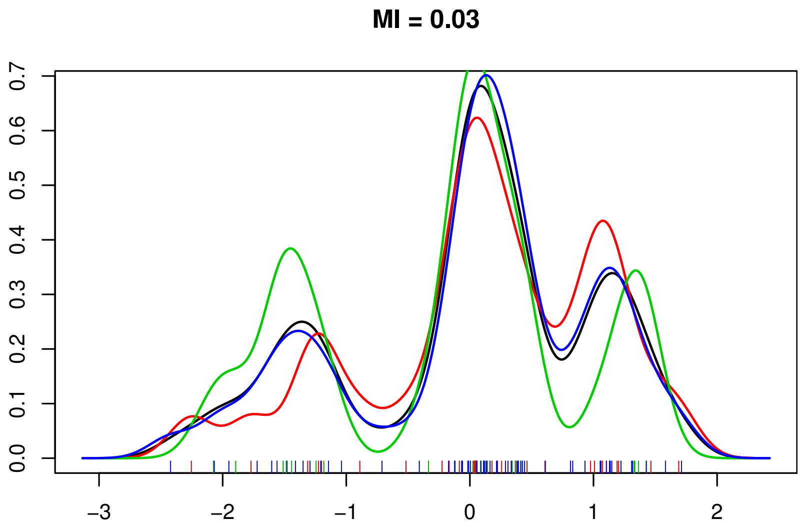


FIGURE 1: Interpretation of high and low MI scores: (a) shows the combined ($f(y)$, black) and conditional ($f(y \mid x_i)$, coloured) densities when there is a strong association between continuous and discrete variables; (b) shows a low MI resulting from a random permutation of the grouping variable. This demonstrates how greater divergences $D(f(y \mid x_i) \parallel f(y))$ correspond to greater MI.

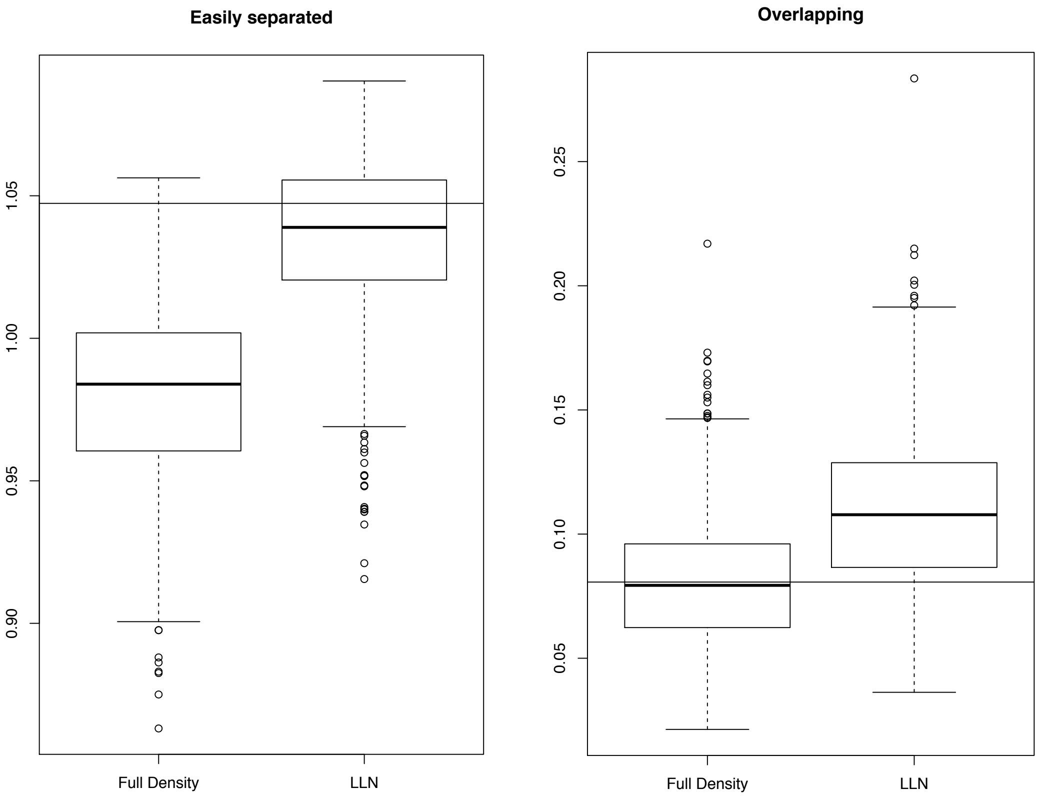


Figure 2: Comparison of MI estimators for easily separable and overlapping simulated data

6 Computational issues

Computational tractability influences our analysis decisions. Although `R` is a common language choice for statistical and machine learning researchers, it is an interpreted language that is known to be comparatively slow for iterative procedures. Code that makes use of built-in vectorised functions runs substantially faster, with further improvement through the use of a fast linear algebra library. We use the Automatically Tuned Linear Algebra

System (ATLAS) as a replacement for R's non-optimised Basic Linear Algebra System (BLAS) implementation. The ATLAS compilation process includes a series of benchmarks to choose the fastest computational kernels for a given system. We perform timings on an HP desktop with Intel Core2 Duo E8400 3 GHz CPU, 4 Gb RAM running OpenSUSE 11.3 and R 2.12.1 built against multi-threaded ATLAS 3.8.3 dynamic libraries.

6.1 Interfacing R with C

Computational bottlenecks are identified using the `Rprof()` profiling function and replaced with dynamically linked `C` code. Two `R` functions are available for this: `.C()` and `.Call()`. The function `.C()` allows `C` functions to be written that refer to `R` objects via pointers. The function `.Call()` allows the direct manipulation of `R` objects using their internal representation (as S-Expressions) with direct access to `R`'s memory management and garbage collection. In general we find the `.Call()` interface results in faster code at the expense of increased programming complexity (see Section 6.1.1). It is also possible within `.Call()` to directly access the Fortran BLAS (in our case ATLAS) functions, allowing us to directly translate entire `R` functions to compiled code.

6.1.1 Simpson's rule

We compare four different approaches to numerical integration using Simpson's rule. Our baseline is a pure `R` function making as much use as possible of vectorised functions. This is compared to inlined `C++` code (using the `Inline` and `Rcpp` packages) and `C` code written using either the `.C()` or `.Call()` interface. A vector of `10001` random standard normal values is used as the integrand. Table 1 reports the total time taken to integrate this `10000` times for each of the four methods. The `.C()` interface is used with the `DUP = FALSE` option to avoid unnecessary duplication of objects in memory.

TABLE 1: Speed comparison for 10000 calculations of Simpson's rule using four different implementations

Method	Time (Seconds)	Speedup
Pure R	41.10	1
Inline C++	5.42	8
`.C()`	0.53	78
`.Call()`	0.36	114

TABLE 2: Three implementations of the vector dot product for short and long vectors

Vector length	Method	Time (Seconds)	Speedup
Long ($n = 10001$)	`x %*% y`	9.81	1.0
	`crossprod(x,y)`	3.12	3.1
	`.Call()` + BLAS	0.93	10.5
Short ($n = 101$)	`x %*% y`	0.18	1.0
	`crossprod(x,y)`	0.21	0.9
	`.Call()` + BLAS	0.27	0.7

6.1.2 Calling BLAS/ATLAS directly from within `.Call()`

Vectorised functions are faster in R due to the use of BLAS/ATLAS libraries. To check that this is still the case when called directly from C code we generate two vectors of 10001 random standard normal values x and y and compare the time taken to calculate the dot product $x \cdot y$ 100000 times (Table 2). We compare the `%*%` operator with the `crossprod()` function and a C function written to make use of Fortran BLAS calls. For the long vectors x and y using BLAS within `.Call()` is substantially faster. However we find that interpreted R code runs faster if x and y are short, say 101 elements in length (Table 2).

6.2 Mutual information estimators

Our implementations are influenced by the above results. The inclusion of the three different types of comparison (discrete, continuous, and mixed) results in the block MI matrix

$$\mathsf{MI} := \begin{bmatrix} \mathsf{A} & \mathsf{B} \\ \mathsf{B}' & \mathsf{C} \end{bmatrix} .$$

Block A holds the continuous versus continuous comparisons, B holds the continuous versus discrete comparison and C holds the discrete versus discrete comparisons. Block B is estimated using the methods of Sections 3.1 and 3.2. Block C is estimated using a straightforward approach implemented with `.C()`. We give timings for calculating A, B and C on the full F2 intercross dataset, in detail:

A We estimate A using a single bandwidth chosen to be the average of all `dpik()` density estimation bandwidths for the continuous variables. We then estimate A with the vectorised version of the fast algorithm of Qiu et al. [7] and compare performance between a pure `R` implementation and complete translation into `C` using `.Call()`. Pure `R` code finds A in 2.05 minutes whereas a complete translation into `.Call()` requires 1.79 minutes.

B We estimate B twice, using either the full kernel density approach (Section 3.2) or the LLN approach (Section 3.1). MI is calculated for each SNP against all genes. A pure `R` implementation of (5) requires 9.15 hours. The full kernel density version is implemented primarily in `R` code with Simpson's rule and one other loop compiled with `C` (and using the `bkde()` function which makes use of Fortran code). This approach requires 3.43 hours (non-optimised pure `R` versions take up to 12 hours).

C We find C by using a direct application of the definition of MI (1) using empirical relative frequencies for each pair of SNPs. Pure `R` takes 12.83 minutes; our `.C()` function takes 5.67 minutes.

7 Application to mouse data

We simplify the approach of Zhang and Horvath [10] by clustering the observed MI matrix with minimal transformations. To ensure that the three comparisons represented by A, B and C are comparable we apply the Context Likelihood of Relatedness (CLR) algorithm of Faith et al. [3] individually to each of the submatrices, before applying CLR again to the final matrix. CLR replaces each MI element with $(z_i^2 + z_j^2)^{1/2}$ where z_i and z_j are row- and column-wise z-scores respectively. This emphasises unusually high levels of association. We then simply apply the robust Partitioning Around Medoids (PAM) clustering algorithm. Choosing the target number of clusters is difficult. At this stage we simply choose 20 groups as this corresponds to the number of chromosomes in mice.

Figure 3 shows the resulting PAM clusters. As in the results of Ghazalpour et al. [5], a group with disproportionate association with weight is clearly visible. The red and blue lines show the upper tertile and quartile of all CLR scores ignoring the clusters. Our technique gives qualitatively similar results despite Ghazalpour et al. [5] only considering the continuous gene expression values for the network and clustering.

8 Conclusion

We developed an approach that uses mutual information to identify continuous variables strongly associated with categorical variables and successfully applied this to a biological dataset. There is a clear group within the data that appears to be associated with weight. This signal is strong enough to be apparent using different procedures as demonstrated by the agreement of our results with those of Ghazalpour et al. [5] even though they only used the gene expression data. By including SNPs in the clustering we extend their approach. We also showed that is it possible to calculate relevant quantities in reasonable

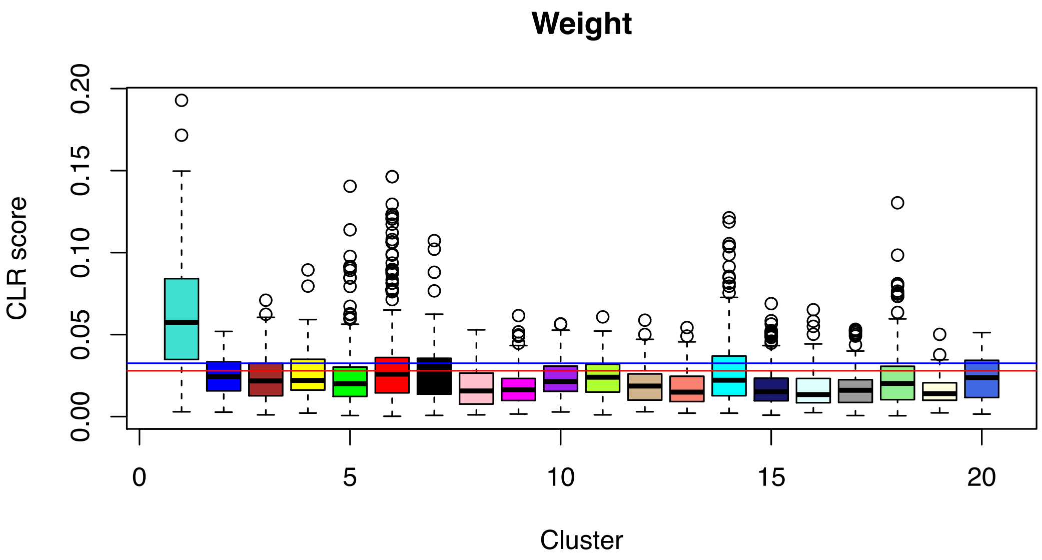


Figure 3: CLR between elements in each cluster and body weight for the F2 mouse intercross data. A CLR is calculated between weight and each SNP or gene. Each boxplot shows these CLR scores within a given cluster. The red and blue lines show the upper tertile and quartile (respectively) of the combined data ignoring clusters.

time by writing the computational kernels in low level `C` called from more user friendly high level `R` functions, outlining some general approaches for improving computational performance.

References

[1] T. M. Cover and J. A. Thomas. *Elements of information theory.* Wiley, 2006. doi:10.1002/0471200611. C712, C713

[2] Z. Dawy, B. Goebel, J. Hagenauer, C. Andreoli, T. Meitinger, and J. C. Mueller. Gene mapping and marker clustering using Shannon's

mutual information. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 3(1):47–56, 2006. doi:10.1109/TCBB.2006.9. C712, C715

[3] J. J. Faith, B. Hayete, J. T. Thaden, I. Mogno, J. Wierzbowski, G. Cottarel, S. Kasif, J. J. Collins, and T. S. Gardner. Large-scale mapping and validation of Escherichia coli transcriptional regulation from a compendium of expression profiles. *PLoS Biol*, 5(1):e8, 2007. doi:10.1371/journal.pbio.0050008. C723

[4] T. F. Fuller, A. Ghazalpour, J. E. Aten, T. A. Drake, A. J. Lusis, and S. Horvath. Weighted gene coexpression network analysis strategies applied to mouse weight. *Mammalian Genome*, 18(6):463–472, 2007. doi:10.1007/s00335-007-9043-3. C712

[5] A. Ghazalpour, S. Doss, B. Zhang, S. Wang, C. Plaisier, R. Castellanos, A. Brozell, E. E. Schadt, T. A. Drake, A. J. Lusis, et al. Integrating genetic and network analysis to characterize genes related to mouse weight. *PLoS Genet*, 2(8):e130, 2006. doi:10.1371/journal.pgen.0020130. C712, C723

[6] Hung T. Nguyen. On modeling of linguistic information using random sets. *Information Sciences*, 34(3):265 – 274, 1984. doi:10.1016/0020-0255(84)90052-5. C713

[7] P. Qiu, A. J. Gentles, and S. K. Plevritis. Fast calculation of pairwise mutual information for gene regulatory network reconstruction. *Computer Methods and Programs in Biomedicine*, 94(2):177–180, 2009. doi:10.1016/0020-0255(84)90052-5. C715, C722

[8] S. J. Sheather and M. C. Jones. A reliable data-based bandwidth selection method for kernel density estimation. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 683–690, 1991. C715

[9] M. P. Wand and M. C. Jones. *Kernel smoothing*. Chapman & Hall/CRC, 1995. C715, C716

[10] B. Zhang and S. Horvath. A general framework for weighted gene co-expression network analysis. *Statistical Applications in Genetics and Molecular Biology*, 4(1):1128, 2005. doi:10.2202/1544-6115.1128. C712, C723

Author addresses

1. **C. Pardy**, Prince of Wales Clinical School, University of New South Wales, New South Wales 2052, Australia.
   `mailto:cpardy@unsw.edu.au`
2. **S. R. Wilson**, Prince of Wales Clinical School, University of New South Wales, New South Wales 2052, Australia.