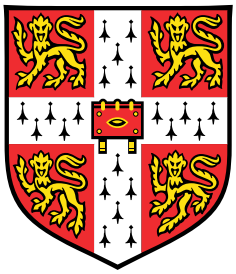
Network Inference Using Independence Criteria



Petras Verbyla

Department of Pure Mathematics and Mathematical Statistics
University of Cambridge

This dissertation is submitted for the degree of
Doctor of Philosophy

Peterhouse July 2018

I would like to dedicate this thesis to my loving mother.

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified bellow.
Chapter 4 is joint work with Nina Desgranges. Nina Desgranges developed the initial version of the Independence Criterion based PC algorithm and did the initial data analysis. I reimplemented the Independence Criterion based PC algorithm for the `R` package and carried out the final testing and data analysis. All other content is my own work.

Petras Verbyla
July 2018

Acknowledgements

I would like to thank Dr. Lorenz Wernisch for all the academic and moral support. For all of our discussions about mathematics and life in general.

Abstract

Biological systems are driven by complex regulatory processes. Graphical models play a crucial role in the analysis and reconstruction of such processes. It is possible to derive regulatory models using network inference algorithms from high-throughput data, for example; from gene or protein expression data. A wide variety of network inference algorithms have been designed and implemented. Our aim is to explore the possibilities of using statistical independence criteria for biological network inference. The contributions of our work can be categorized into four sections.

First, we provide a detailed overview of some of the most popular general independence criteria: distance covariance (dCov), kernel canonical variance (KCC), kernel generalized variance (KGV) and the Hilbert-Schmidt Independence Criterion (HSIC). We provide easy to understand geometrical interpretations for these criteria. We also explicitly show the equivalence of dCov, KGV and HSIC.

Second, we introduce a new criterion for measuring dependence based on the signal to noise ratio (SNRIC). SNRIC is significantly faster to compute than other popular independence criteria. SNRIC is an approximate criterion but becomes exact under many popular modelling assumptions, for example for data from an additive noise model.

Third, we compare the performance of the independence criteria on biological experimental data within the framework of the PC algorithm. Since not all criteria are available in a version that allows for testing conditional independence, we propose and test an approach which relies on residuals and requires only an unconditional version of an independence criterion.

Finally we propose a novel method to infer networks with feedback loops. We use an MCMC sampler, which samples using a loss function based on an independence criterion. This allows us to find networks under very general assumptions, such as non-linear relationships, non-Gaussian noise distributions and feedback loops.

Table of contents

List of figures **xvii**

List of tables **xxiii**

Introduction **1**

1 Background **3**

1.1 Previous Work in Network Inference . . . 3

1.1.1 Correlation and Information Theory Based Methods . . . 3

1.1.2 Boolean Networks . . . 5

1.1.3 Differential and Difference Equations Based Methods . . . 5

1.1.4 Bayesian Networks . . . 6

1.1.5 Feedback Loop Inference . . . 6

1.2 Probabilistic Graphical Models . . . 7

1.2.1 Motivation . . . 7

1.2.2 Graph Theory Definitions . . . 7

1.2.3 Probabilistic Graphical Models . . . 8

1.2.4 Conclusions . . . 13

1.3 PC Algorithm . . . 13

1.3.1 Skeleton Phase . . . 13

1.3.2 Collider Phase . . . 14

1.3.3 Transitive Phase . . . 15

1.4 Loss Function . . . 15

1.5 Kullback-Leibler Divergence . . . 17

1.6 Datasets . . . 18

1.6.1 Simulated Datasets . . . 18

1.6.2 Single-cell Datasets . . . 19

1.6.3 Schistosomiasis Dataset . . . 21

2 Network Inference of Discrete Bayesian Networks 25
2.1 Introduction . . . 25
2.1.1 Network Inference Problem . . . 25
2.1.2 Discrete Bayesian Network Inference . . . 27
2.2 Structure Search . . . 27
2.3 Likelihood . . . 29
2.3.1 Likelihood for Discrete Variable . . . 30
2.3.2 Priors . . . 34
2.3.3 Likelihood Equivalence . . . 34
2.3.4 Likelihood Equivalence Non-Preserving Priors . . . 35
2.4 Missing Value Imputation in the Model . . . 40
2.5 Parallel Tempered MCMC . . . 41
2.5.1 Issues with Chain Mixing . . . 41
2.5.2 Tempered MCMC . . . 42
2.5.3 Parallel Tempered MCMC . . . 43
2.5.4 Temperatures . . . 45
2.6 Results . . . 45
2.6.1 Small Examples . . . 46
2.6.2 11-Variable Examples . . . 48
2.6.3 Conclusions . . . 49
2.6.4 Single-Cell Datasets . . . 51
2.6.5 Schistosomiasis Dataset . . . 52
2.6.6 28-Variable Case . . . 56
2.6.7 Scaling to Larger Networks . . . 57
2.7 Discussion . . . 59
2.7.1 Model Limitations and Future Extensions . . . 59

3 Independence Criteria 61
3.1 Introduction . . . 61
3.1.1 Independence . . . 61
3.2 Distance Correlation . . . 63
3.2.1 Introduction . . . 63
3.2.2 Motivation . . . 63
3.2.3 Definition of dCor . . . 65
3.2.4 Empirical Estimate of dCor . . . 67
3.2.5 Theoretical Justification of dCor . . . 70
3.3 Kernel Canonical Correlation . . . 70

3.3.1 Introduction . . . 70
3.3.2 Motivation . . . 71
3.3.3 Reproducing Kernel Hilbert Space . . . 72
3.3.4 Definition of the $\mathcal{F}$-correlation . . . 77
3.3.5 Empirical Estimate of the $\mathcal{F}$-correlation . . . 78
3.3.6 Kernel Generalized Variance . . . 81
3.4 Hilbert-Schmidt Independence Criterion . . . 84
3.4.1 The Hilbert-Schmidt Space . . . 84
3.4.2 Hilbert-Schmidt Independence Criterion . . . 86
3.4.3 Empirical Estimate of HSIC . . . 87
3.4.4 Asymptotic Results . . . 88
3.5 Similarities between the Independence Criteria . . . 91
3.5.1 Relationship between dCov and HSIC . . . 92
3.5.2 Relationship between KGV and HSIC . . . 95
3.6 Statistical Tests of Independence . . . 98
3.6.1 Introduction . . . 98
3.6.2 Test of the Unconditional Independence . . . 98
3.6.3 Test of the Conditional Independence . . . 100
3.6.4 Simulation Results for the Independence Tests . . . 102
3.7 Signal to Noise Ratio Independence Criterion . . . 105
3.7.1 Introduction . . . 105
3.7.2 Motivation . . . 106
3.7.3 Signal to Noise Ratio Independence Criterion . . . 110
3.7.4 SNRIC and the General Independence . . . 111
3.7.5 Definition of the SNRIC . . . 113
3.7.6 SNR Independence Test . . . 113
3.7.7 Unconditional Independence Test Examples . . . 115
3.7.8 Counter Example to SNRIC . . . 121
3.7.9 SNRIC Conditional Independence Test . . . 122
3.7.10 Proofs . . . 123
3.8 Conclusions . . . 125
3.8.1 Future Work . . . 126

4 Independence Criteria Based PC Algorithm 127
4.1 Introduction . . . 127
4.2 Kernel PC . . . 127
4.3 Inferring Directionality Using the Independence Criteria . . . 129

4.4 Results . . . 133
4.4.1 Comparison of Network Inference Algorithms . . . 133
4.4.2 Data Simulated from an Artificial Network . . . 134
4.4.3 Data Simulated by Re-sampling . . . 134
4.4.4 Original Data . . . 139
4.4.5 Combining All Datasets . . . 140
4.4.6 Discovering Directions . . . 140
4.4.7 Schistosomiasis Dataset . . . 142
4.5 Discussion . . . 143
4.5.1 Model Limitations . . . 146
4.5.2 Future work . . . 147

5 MCMC Sampling Using Loss Function 149
5.1 Introduction/Motivation . . . 149
5.2 Model . . . 149
5.2.1 Structural Equation Model . . . 150
5.2.2 Additive Noise Model . . . 151
5.3 Theory . . . 153
5.4 Reversible Jump MCMC . . . 153
5.4.1 Jump to a Higher Dimensional Space . . . 153
5.4.2 Centring Proposals . . . 155
5.4.3 Weak Non-Identifiability Approach . . . 155
5.4.4 The Conditional Maximization Approach . . . 156
5.4.5 Zeroth-Order Method . . . 156
5.4.6 Example 1: Weak Non-Identifiability . . . 157
5.4.7 Example 2: Conditional Maximization . . . 158
5.4.8 Jump to a Lower Dimensional Space or Removing an Edge . . . 158
5.4.9 Inverting an Edge . . . 159
5.5 Sampling from the Loss Function . . . 160
5.6 Algorithm . . . 162
5.6.1 Introduction . . . 162
5.6.2 Loss Function Based on Independence Criterion . . . 163
5.6.3 Linear Model . . . 164
5.6.4 Piecewise Linear Model . . . 165
5.6.5 Prior . . . 166
5.6.6 Sampling Distribution . . . 167
5.6.7 Hyperparameters . . . 168

5.6.8 Sampler . . . 170
5.7 Results . . . 170
5.7.1 Simulating Data with Cycles . . . 170
5.7.2 Simulated Examples . . . 171
5.7.3 Single-Cell Data . . . 179
5.7.4 Schistosomiasis Dataset . . . 181
5.8 Discussion . . . 182
5.8.1 Model Limitations . . . 185
5.8.2 Future Work . . . 188

Conclusions **189**

References **193**

Appendix A Data **201**

Appendix B Discrete Bayesian Network **203**

Appendix C Independence Criteria **205**
C.1 $\mathcal{F}$-correlation . . . 205
C.1.1 Regularization of the $\mathcal{F}$-correlation . . . 206
C.1.2 Kernel Generalized Variance . . . 208
C.1.3 HSIC . . . 211
C.2 U-statistics . . . 219
C.3 Approximating Mutual information using Gaussian random variables . 223

Appendix D Kernel PC **225**

List of figures

1.1 Example of an undirected edge (on the left) and a directed edge (on the right). 7
1.2 Example of the parents (on the left) and the children (on the right) of a node. 8
1.3 Example of a DAG (on the left) and a graph that is not a DAG (on the right). 8
1.4 Examples of the chain (on the left), the fork (in the middle) and the collider (on the right) graphs. 9
1.5 Example of a probabilistic graphical model. 10
1.6 Examples of three graphs with the same skeleton and same d-separations. 12
1.7 Collider phase. $Z \notin \text{sepset}(X, Y)$. 14
1.8 Meek's rules. 15
1.9 Summary of known dependencies (after Sachs et al. (2005)). 20
1.10 Experimental dataset 8 from Sachs et al. (2005) after log-transformation. 20
1.11 Schistosomiasis dataset after the log-transformation 23

2.1 Example of underlying true network that leads to undirected edge. . . . 26
2.2 Graph G of the Bayesian Network. 33
2.3 On the left: bimodal distribution $p(x)$; on the right: tempered distribution $q(x) \propto p(x)^{\frac{1}{10}}$, with temperature 10 42
2.4 Example of running 5 chains at temperatures $t_1, ..., t_5$ for 6 sweeps . . 44
2.5 Adjacency matrix for the original graph G_{Ch} and the output of MCMC sampler. Matrix M element m_{ij} represents the probability with which the i^{th} node is a parent of the j^{th} node. 47
2.6 Adjacency matrix for the original graph G_T and the output of MCMC sampler. 48
2.7 Adjacency matrix for the original graph G_C and the output of MCMC sampler. 49

2.8 Adjacency matrix for the original graph G_{Tr} and the output of MCMC sampler. 50
2.9 Adjacency matrix for the original graph on 11 nodes G_1 and the output of MCMC sampler. 50
2.10 Adjacency matrix for the original graph on 11 nodes G_2 and the output of MCMC sampler. 51
2.11 Adjacency matrix for the original graph on 11 nodes G_3 and the output of MCMC sampler. 51
2.12 Results from datasets simulated from the Sachs Dataset 8. 52
2.13 Results from the Sachs Dataset 8. 53
2.14 Network from the single-cell data generated by the MCMC using the discretized data. Green are the correctly identified edges, red are wrongly identified edges and dashed black are the edges that the algorithm did not find. Solid lines represent edges with high probability and dashed lines represent edges with low probability. 53
2.15 The Schistosomiasis network adjacency matrix found by our algorithm using Gibbs sampling for the missing value imputation. Matrix M element m_{ij} represents the probability with which the i^{th} node is a parent of the j^{th} node. 54
2.16 The Schistosomiasis network adjacency matrix found by our algorithm using the MICE package for the missing value imputation. Matrix M element m_{ij} represents the probability with which the i^{th} node is a parent of the j^{th} node. 55
2.17 Network for the Schistosomiasis dataset; only the edges with probability above 0.5 are shown. 55
2.18 The Schistosomiasis network adjacency matrix found by our algorithm using the MICE package for the missing value imputation. Matrix M element m_{ij} represents the probability with which the i^{th} node is a parent of the j^{th} node. 56

3.1 Example of truly independent samples. 62
3.2 Original data divided into 4 quadrants. 65
3.3 Distances of the data from Figure 3.2. The colour of Section represents the distances along the same coloured arrow in Figure 3.2, for example the green section represents the distances between two quadrants both at the top or both at the bottom, while the black section represents the distances inside the quadrant. 65

3.4 Example of data that is dependent but has zero correlation and a function that allows us to extract a linear dependence from this data. . 72
3.5 Data from Figure 3.4a with function f from the Figure 3.4b applied to one and both variables. 72
3.6 Data simulated with nonlinear dependencies 103
3.7 Dependency of p-values of HSIC and dCov tests on varying parameters, kernel width λ for HSIC and index ξ for dCov. 104
3.8 Time efficiency of independence tests. 105
3.9 Dependency of the p-values of the HSIC Gamma test on a varying kernel width parameter λ, noise level ϵ and the number of observations. Model: $Y \sim \sin(X) + \mathcal{N}(0, \sigma^2)$, $X \sim \mathcal{U}(0, 10)$. 106
3.10 Summary of known dependencies with high (green) and low (gray) SNR. 111
3.11 CDF for SNR approximation using a Gamma distribution 115
3.12 Three examples of data with linear dependency and varying noise. . . . 116
3.13 Three examples of data with nonlinear dependency and varying noise. . 117
3.14 Three examples of data with circular dependency and varying noise. . . 118
3.15 Three examples of independent data after varying rotations applied. . . 119
3.16 Three examples of data with the noise level dependency and varying noise. 120
3.17 Counterexample for SNRIC. 121

4.1 Model with a non-linear relationship and Gaussian noise. 130
4.2 Model with a linear relationship and a non-Gaussian noise. 130
4.3 Model with a linear relationship and a Gaussian noise. 131
4.4 Model with an unobserved latent variable. 132
4.5 Multiplicative noise model. 133
4.6 Toy simulated example on 9 nodes and 300 observations. 135
4.7 ROC curves to compare kPCs, dPC, SNR-PC and PC algorithms on the toy simulated example. The solid lines are the mean ROC curve and the dotted lines are the mean ROC curve ± one standard deviation. 135
4.8 Data simulated with non-reduced noise from dataset 8. 137
4.9 ROC curve to compare kPCs, dPC, SNR-PC and PC algorithms on the data simulated with reduced noise from dataset 8. 138
4.10 ROC curves for dataset 8. 139
4.11 All 8 datasets combined. 140

4.12 Output of the kPC-Resid algorithm on the data simulated from dataset 8. Colour coding: dashed black undirected or doubly directed edges represent correctly identified undirected edges, green directed edges represent correct, while red directed edges represent incorrect orientations. Dashed black oriented edges are from the previous phase. 142
4.13 Output of the dPC algorithm on the data simulated from the dataset 8. 142
4.14 Network for the Schistosomiasis dataset; only the edges present in all four kPC variants with $\alpha = 0.1$ are present. 143
4.15 The Schistosomiasis network adjacency matrix found by kPC algorithm using various independence criteria. Matrix M element m_{ij} represents the proportion of outcomes in which the i^{th} node is a parent of the j^{th} node. 144

5.1 Example of a Structural Equation Model. 151
5.2 Example of a piece wise linear function. 165
5.3 The best solutions for different parameters. 169
5.4 Small example with a cycle. 173
5.5 Small example with 4 nodes and 4 edges, containing a cycle. Red solid line represents the true parameter value, red dotted line the mean MCMC sample value. 173
5.6 MCMC simulation for Figure 5.4 using 3 knots. 174
5.7 Second small example with a cycle. 174
5.8 Small example with 6 nodes and 8 edges, containing a cycle. Red solid line represents the true parameter value, red dotted line the mean MCMC sample value. 175
5.9 Example with 6 nodes and 7 edges, containing a "big" cycle of 4 nodes: $X_3 \to X_4 \to X_5 \to X_6 \to X_3$. 175
5.10 MCMC simulation for Figure 5.9. Red line represents the true parameter value, red dotted lines the mean MCMC sample value. 176
5.11 Example with 7 nodes and 11 edges, containing two cycles: $X_3 \leftrightarrow X_4$ and $X_5 \leftrightarrow X_6$. 177
5.12 MCMC simulation for Figure 5.11. Red line represents the true parameter value, red dotted lines the mean MCMC sample value. 178
5.13 Second small example with a cycle and non-linear relationship. 178
5.14 MCMC simulation for Figure 5.11. Solid lines represents the true parameter values, dotted lines the mean MCMC sample values. 179

5.15 ROC curves for the datasets simulated from the single-cell dataset 8. MCMC with loss function using linear relationships. 180
5.16 Network for the datasets simulated from the single-cell dataset 8. Green are the correctly identified edges, red are wrongly identified edges and dashed black are the edges that algorithm did not find. 181
5.17 ROC curves for the single-cell dataset 8. MCMC with loss function using non-linear relationships. 181
5.18 ROC curves for one dataset simulated from the single-cell dataset 8. MCMC with loss function using non-linear relationships. 182
5.19 The Schistosomiasis network adjacency matrix found by our algorithm. Matrix M element m_{ij} represents the probability that the i^{th} node is a parent of the j^{th} node. 183
5.20 The Schistosomiasis network adjacency matrix found by our algorithm. Matrix M element m_{ij} represents the size of the effect of the i^{th} node on the j^{th} node. 184
5.21 Network for the Schistosomiasis dataset; only the edges with probability above 0.5 and parameters above 0.1 in absolute value are present. . . . 185

A.1 Data simulated from the single-cell dataset 8 after the log - transformation (from Sachs et al. (2005)). 201

B.1 Discrete MCMC effectiveness on data simulated from the single-cell data, all eight datasets. 204

D.1 ROC curves to compare kPCs, dPC, SNR-PC and PC algorithms on the data simulated from single-cell data from Sachs et al. (2005). 226
D.2 ROC curves to compare kPCs, dPC, SNR-PC and PC algorithms on all 8 datasets from Sachs et al. (2005). 227

List of tables

1.1 Chapters and types of networks 19
1.2 Table of 8 single-cell datasets provided in Sachs et al. (2005). 21
1.3 Schistosomiasis dataset; variables used in the initial model 23

2.1 Probabilities of the Bayesian Network. 33
2.2 Data generated from the Bayesian Network. 33
2.3 Time taken and the mean Area Under the Curve achieved by the BN MCMC algorithm for networks with various number of nodes and various number of observations. 58

3.1 Testing independence criteria. All the p-value estimates are the mean of 100 p-values from repetitions for each of the four tests. The size of the simulated sample is 300. 102
3.2 Relationship between independence criteria and the SNR's for the linear relationship with varying noise levels. 108
3.3 Relationship between independence criteria and the SNR's for the non-linear relationship with varying noise levels. 108
3.4 SNR_1 for all 8 datasets combined. 110
3.5 SNR_2 for all 8 datasets combined. 111
3.6 Combinations of distributions used to generate Figure 3.11. 115
3.7 The p-value estimates on the data with linear dependence and varying noise. 117
3.8 The p-value estimates on the data with the non-linear dependence and varying noise. 118
3.9 The p-value estimates on the circular dependency data with varying noise. 118
3.10 The p-value estimates on the independent data after varying rotation. . 119
3.11 The p-value estimates on the data with the noise level dependency and varying noise. 120

3.12 The p-value estimates on the counter example for SNRIC data. 122

4.1 The mean p-values of the residuals of the regressions x on y and y on x being independent in a model with non-linear relationship with a Gaussian noise. 129
4.2 The mean p-values of the residuals of the regressions x on y and y on x being independent in a model with a linear relationship with a non-Gaussian noise. 130
4.3 The mean p-values of the residuals of the regressions x on y and y on x being independent in a model with a linear relationship with a Gaussian noise. 131
4.4 The mean p-values of the residuals of the regressions x on y and y on x being independent in a model with an unobserved latent variable. . . . 131
4.5 The mean p-values of the residuals of the regressions x on y and y on x being independent in a model that is not an additive noise model. . . . 132
4.6 Free parameters for kPCs and dPC. 134
4.7 Comparison of the PC algorithm versions on a small simulated example. Number in the i^{th} row and the j^{th} column represents how many times algorithm i outperformed the algorithm j. 136
4.8 Comparison of the PC algorithm versions on the data simulated from dataset 8. 138
4.9 Comparison of the PC algorithm versions on all 8 simulated datasets. . 139
4.10 The average number of correctly and wrongly oriented edges as well as not oriented edges for the algorithms. 141

5.1 Time taken to evaluate the SNR IC based loss function. Rows are the number of observations, columns are the number of nodes. 186
5.2 Time taken to evaluate the dCov based loss function. Rows are the number of observations, columns are the number of nodes. 187
5.3 Time taken to evaluate the HSIC based loss function. Rows are the number of observations, columns are the number of nodes. 187

Introduction

Biological systems are driven by complex regulatory processes. Graphical models play a crucial role in the analysis and reconstruction of such processes. It is possible to derive regulatory models using network inference algorithms from high-throughput data, for example, from gene or protein expression data. A wide variety of network inference algorithms have been designed and implemented and necessitate common platforms for assessment, for example, the DREAM network inference challenges (Marbach et al., 2012), to provide objective means for choosing reliable inference algorithms.

Inference algorithms are based on a variety of statistical principles. However, most rely on some form of estimating or testing the similarity or correlation between genes, for example, GeneNet (Opgen-Rhein and Strimmer, 2007) and MutRank (Obayashi and Kinoshita, 2009), or on mutual information as does CLR (Faith et al., 2007) and RelNet (Butte and Kohane, 2000), or on regression and feature selection as does MRNET (Meyer, 2010; Meyer et al., 2007) and Genie3 (Huynh-Thu et al., 2010).

The use of general independence criteria for network inference has been suggested in contexts outside biological networks (Lippert et al., 2009). Our aim is to explore the possibilities of using general independence criteria for biological network inference.

The contributions of our work can be categorized into four sections. First, we provide a detailed overview of general independence criteria: distance covariance (dCov) Székely et al. (2007), kernel canonical variance (KCC), kernel generalized variance (KGV) (Bach and Jordan, 2002) and the Hilbert-Schmidt Independence Criterion (HSIC) (Gretton et al., 2005). We provide an easy to understand geometrical intuition behind these criteria. Sejdinovic et al. (2013) showed a general equivalence between distance (for example dCov) and Reproducing Kernel Hilbert Spaces (for example HSIC) based independence criteria. We explicitly show that dCov and KGV are equivalent to HSIC, which provides more insight into the nature of the equivalence than the general result. Second, we introduce a new criterion for measuring dependence based on the signal to noise ratio (SNRIC). SNRIC is significantly faster than the previously mentioned general independence criteria. SNRIC is an approximate criterion for dependence, but

we show that in the special case of data coming from an additive and multiplicative noise model, it is exact. Third, we compare the performance of the previously discussed independence criteria on biological experimental data within the framework of the PC algorithm. Since not all criteria are available in a version that allows for testing conditional independence, we propose and test an approach which relies on residuals and requires only an unconditional version of an independence criterion. The true network is rarely known when assessing algorithms. Hence, we also propose a simulation method that, starting from experimental data and a target network, produces simulated data according to the dependency structure of the target network but which are otherwise as close to the original data as possible in their noise characteristics and functional (possibly non-linear) forms of dependencies. We make all algorithms and data available as a package `kpcalg` (Verbyla et al., 2017) for the `R` statistical environment (R Core Team, 2014). Finally we propose an MCMC sampler, which samples from a loss function based on an independence criterion. This is a very flexible method which performs well on networks with non-linear relationships, non-Gaussian noise terms and cyclic relationships.

This thesis is organised as follows. In Chapter 1 we introduce the general framework of the probabilistic graphical models and the datasets which will be used throughout this work. In Chapter 2 we discuss network inference using an MCMC sampler. In Chapter 3 we review general independence criteria and introduce SNRIC. In Chapter 4 we compare statistical tests for independence based on different independence criteria within the framework of the PC algorithm. In Chapter 5 we introduce the MCMC sampler using a loss function based on an independence criterion.

Chapter 1

Background

In this chapter we provide the required background to make this thesis as self-contained as possible. We begin by discussing the work that has already been done in the network inference. We discuss the most popular classes of inference algorithms, their advantages and limitations. Then we provide the required theoretical background: we discuss probabilistic graphical models that will be used throughout this thesis, as well as PC algorithm (used in Chapter 4), loss function and Kullback-Leibler divergence (used in Chapter 5). Finally we introduce datasets that we will be using in this thesis.

1.1 Previous Work in Network Inference

Network inference is increasing in popularity and therefore an increasing number of methods is being developed. In this section we discuss the most popular approaches used in the field. We summarize previous reviews on the network inference models Hecker et al. (2009), Noor et al. (2013) and Banf and Rhee (2017). These reviews define following broad categories of network inference algorithms: Correlation and Information Theoretical methods, Boolean networks, Bayesian networks and Differential and Difference Equation methods. We present each category in more detail as well as their advantages and limitations.

1.1.1 Correlation and Information Theory Based Methods

Methods in this class use correlation (Pearson's , Spearman's (Usadel et al., 2009) or weighted (Zhang and Horvath, 2005)) based or information theoretical (mutual information, conditional mutual information or three way mutual information) value function to determine a weight (importance) for each edge and removes edges with

low weights (unimportant edges). These methods are relatively simple and have a low computational cost therefore it is a good choice to deal with large networks. On the other hand these methods tend to be symmetric so edges should be only interpreted as showing co-regulation of genes rather than causal relationships. This is especially important when considering cyclic relationships. We provide a short list of the most relevant methods.

- RELNET (RELevance NETworks) (Butte and Kohane, 2000) starts from a fully connected graph, assigns each edge a weight corresponding to the mutual information between its nodes and removes edges that fall below a certain threshold.
- Mutual Information network (Steuer et al., 2002) is a simple algorithm that keeps edges that correspond to pairs of genes with high mutual information.
- Correlation network (Stuart et al., 2003) is a simple algorithm which keeps edges that correspond to highly correlated (above a predetermined threshold) pairs of genes.
- ARACNE (Algorithm for the Reverse engineering of Accurate Cellular NEtworks) (Basso et al., 2005; Margolin et al., 2006) instead of considering independent edges, considers triples of nodes removing the edge with smallest mutual information score, assuming that it is most likely to be an indirect relationship.
- CLR (Context Likelihood of Relatedness) (Faith et al., 2007) works similarly to other information theoretical network inference methods but considers a specially tailored threshold for each gene pair rather than a global threshold.
- Soranzo et al. (2007) use conditional mutual information.
- MI3 (Luo et al., 2008) use three way mutual information.
- MRNET (minimum redundancy, maximum relevance) (Meyer, 2010; Meyer et al., 2007) chooses the best set of predictors for each gene individually.
- C3NET (conservative causal core) (Altay and Emmert-Streib, 2010) keeps only the edge with highest mutual information for each gene.

1.1.2 Boolean Networks

Boolean networks use binary data where each gene in the network can be in a state "on" or "off". Boolean networks are directed graphs where each edge is a Boolean function consisting of logical operations "AND", "OR", "NOT". Advantage of the boolean networks is their interpretability. On the other hand using only two states is usually not sufficient to represent genes accurately, so we incur information loss. Boolean networks are time dependent and gene expression time series often have few time points, which makes network reconstruction difficult. Example of a boolean network is REVEAL (REVerse Engineering ALgorithm) by Liang et al. (1998).

1.1.3 Differential and Difference Equations Based Methods

Idea behind these methods is to use ordinary differential equations (ODEs) to express the relationships between genes. Simplest approach is to use linear ODEs. Approximating linear ODEs as linear difference equations allows us to transform the problem to a system of linear algebraic equations. Linear models yield good results if data comes from a steady state of the system, otherwise results might be unstable (Filkov, 2005). Linear models often might be insufficient to explain complex biological systems (Savageau, 1970) therefore non-linear models may be more appropriate. The shortcoming of the non-linear models is that they have significantly more parameters than the linear counterparts and the sample size of observations is often too small to reliably identify these parameters. Some of the most popular methods are:

- Random forest (Breiman, 2001) for each gene a number of decision trees are being grown and final result is an averaging over several trees
- NIR (Network Identification by multiple Regression) (Gardner et al., 2003) use linear difference equations on steady state RNA expression measurements
- MNI (Microarray Network Identification) (di Bernardo et al., 2005) use linear difference equations on steady state RNA expression measurements
- TSNI (Time-Series Network Identification) (Bansal et al., 2006) use linear difference equations on time series RNA expression measurements
- genie3 (Huynh-Thu et al., 2010) tree-based ensemble regression method
- TIGRESS algorithm (Haury et al., 2012) uses stability selection procedure to produce more consistent results.

1.1.4 Bayesian Networks

Network is represented by a directed acyclic graph G where each variable X_i is drawn from conditional probability distribution $\Pr(X_i \mid \mathrm{Pa}(X_i))$, where $\mathrm{Pa}(X_i)$ is the set of parents of X_i. Bayesian network inference techniques are covered in Heckerman (1996) and Needham et al. (2007). Probabilistic framework is convenient to represent causal relationships (Aluru, 2006). Disadvantages of Bayesian networks is that the number of possible networks increase super-exponentially with the number of nodes, therefore it is not feasible to compute the likelihood of all possible networks. To solve this problem approximate sampling using Markov Chain Monte Carlo (MCMC) may be used (Tasaki et al., 2015). Another issue is that the Bayesian network output is a directed acyclic graph, i.e. cyclic relationships are not allowed. To circumvent this issue dynamical Bayesian networks (Perrin et al., 2003; Van Berlo et al., 2003) were proposed.

1.1.5 Feedback Loop Inference

Regulatory feedback loops frequently occur in biological networks (Mangan and Alon, 2003; Marbach et al., 2012, 2010). Therefore it is important to have models that are able to capture this important feature. Correlation and information theoretical methods are symmetrical in nature and therefore cannot be easily used to infer feedback loops. Asymmetric version of mutual information used by Rao et al. (2007) allows to obtain directed networks that could be used to infer cyclic relationships. Boolean and Bayesian networks use acyclic graph structure and therefore cannot deal with feedback loops by construction. A workaround in the Bayesian network case is dynamical Bayesian network, though it requires time series data. Differential equation based models consider each gene separately and therefore may miss the feedback loops.
There has been some work done to specifically tackle the problem of cyclic relationships. First provably correct algorithm to discover general linear directed graphs is Richardson's Cyclic Causal Discovery algorithm (Richardson, 1996). Lacerda et al. (2008) proposes to use independent component analysis (ICA) to discover cyclic causal models. It builds on the work of Shimizu et al. (2006) and uses linear non-Gaussian structural equation models. Hyttinen et al. (2010) and Antti Hyttinen (2012) also use structural equation model to infer feedback loops. This approach assumes linear relationships between variables, we discuss it in more detail in Section 5.2.1. In Chapter 5 we build on their work by relaxing the linearity assumption.

1.2 Probabilistic Graphical Models

1.2.1 Motivation

A Probabilistic Graphical Model is an efficient way of putting the knowledge about a domain into a mathematical framework. It lies at the intersection of probability and graph theory and can therefore benefit from results of both fields. In this section we describe the mathematical framework of probabilistic graphical models. We start with basic concepts from graph theory. We continue by introducing graphical models, and argue why they are useful in analysing experimental datasets.

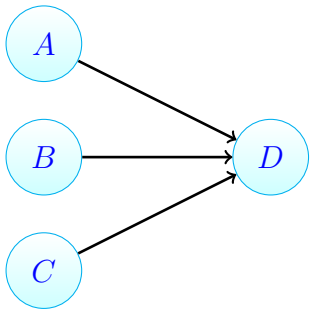
1.2.2 Graph Theory Definitions

Here we provide some essential definitions from graph theory and some examples. Definitions are taken mainly from Murphy (2012) and Koller and Friedman (2009).
A *graph* $G = (V, E)$ consists of a set of nodes (or vertices), $V = \{v_1, ...v_n\}$, and a set of edges $E = \{(s, t) : s, t \in V\} \subseteq V \times V$. An edge between two nodes v_1 and v_2 can be either undirected, symbolically $v_1 - v_2$, or directed, symbolically $v_1 \to v_2$, see Figure 1.1 for an example.

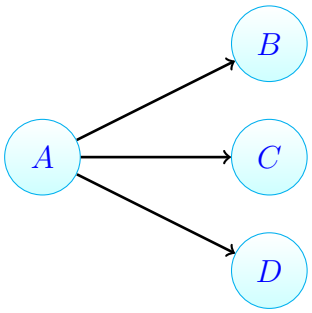
(a) Undirected edge. (b) Directed edge.

Fig. 1.1 Example of an undirected edge (on the left) and a directed edge (on the right).

We say that the graph is *directed*, if all its edges are directed. For any graph we define its *adjacency matrix* M, to be an $n \times n$ square matrix such that $M(u, v) = 1$ iff $(u, v) \in E$. The *parents* of a node $v \in V$ is the set of all the nodes that feed into it, i.e. $\text{Pa}_G(v) = \{u : (u, v) \in E\}$ and the *children* of a node $u \in V$ is the set of all the nodes that feed out of it, i.e. $\text{Ch}_G(u) = \{v : (u, v) \in E\}$. Examples of parents and children are provided in Figure 1.2.
For a directed graph, a *path* $u \rightsquigarrow v$ is a series of directed edges leading from u to v, i.e. $u \rightsquigarrow v = (v_1, v_2, ..., v_k)$ s.t. $v_1 = u$, $v_k = v$ and $(v_i, v_{i+1}) \in E$, for all $i = 1, 2, ...k - 1$. If $v_1 = v_k$ and $k \geq 2$ we say that the path $(v_1, ..., v_k)$ is a *cycle* or a loop. A *directed acyclic graph* or *DAG* is a directed graph with no directed cycles. In Figure 1.3 we have two examples of graphs: the one on the left is a DAG, the one on the right is not (as it has a directed cycle (A, C, D, B, A)).

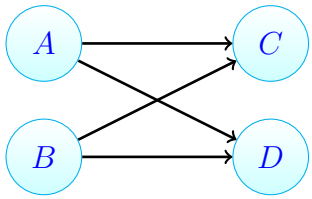


(a) Parents: $\text{Pa}_G(D) = \{A, B, C\}$.

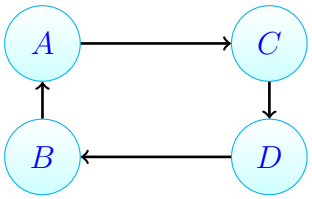


(b) Children: $\text{Ch}_G(A) = \{B, C, D\}$.

Fig. 1.2 Example of the parents (on the left) and the children (on the right) of a node.



(a) Directed Acyclic Graph.



(b) Directed Cyclic Graph.

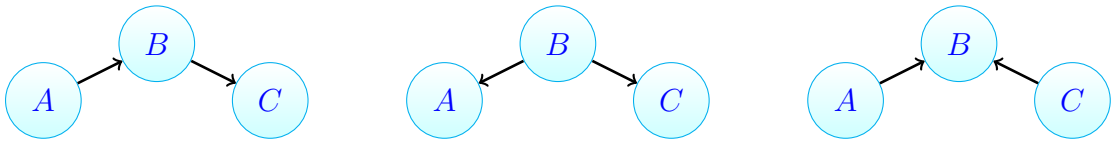
Fig. 1.3 Example of a DAG (on the left) and a graph that is not a DAG (on the right).

For a DAG, a *topological ordering* or *total ordering* is a numbering of nodes such that parents have lower numbers than their children. The *skeleton* of a graph $G = (V, E)$ is a graph with the same set of vertices and the same edge structure, but all edges undirected.
Lets consider a graph $G = (\{A, B, C\}, \{(A, B), (B, C)\})$ on three nodes, i.e the skeleton of graph G is $A - B - C$. Then there are three possible edge orientations. A graph $G = (\{A, B, C\}, \{(A, B), (B, C)\})$ is called

- a *chain* if it has edge orientation $A \leftarrow B \leftarrow C$ or $A \rightarrow B \rightarrow C$. Example in Figure 1.4a.
- a *fork* if it has edge orientation $A \leftarrow B \rightarrow C$. Example in Figure 1.4b.
- an *invert fork* (or *v-structure*) if it has edge orientation $A \rightarrow B \leftarrow C$. In this case the node Y is called a collider. Example in Figure 1.4c.

1.2.3 Probabilistic Graphical Models

The Probabilistic Graphical Model is a compact, graph based representation of the joint probability distribution. Suppose we have a random variable $X = \{X_1, X_2, ..., X_N\}$.



(a) A chain. (b) A fork. (c) A collider.

Fig. 1.4 Examples of the chain (on the left), the fork (in the middle) and the collider (on the right) graphs.

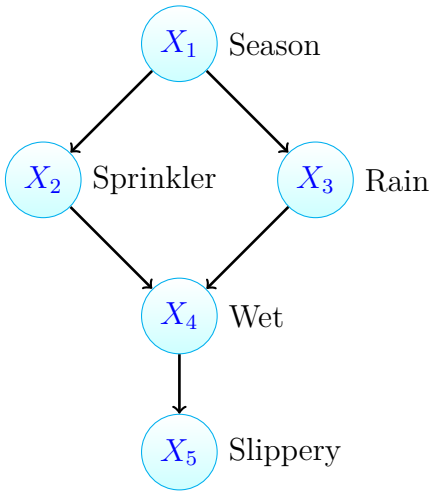
We can use a directed graph $G = (V, E)$ to represent the joint probability distribution of X. We start by defining V, the set of nodes of G as the index set of X, i.e. every random variable X_i for $i = 1, 2, ..., N$ represents a node in the graph. Given a topological ordering we define *ordered Markov property* to be the assumption that a node only depends on its immediate parents, i.e. $u \perp \text{Anc}_G(u) \backslash \text{Pa}_G(u) \mid \text{Pa}_G(u)$, where $\text{Anc}_G(u)$ are the predecessors of node u (Murphy, 2012). Then we say that there is a directed edge from the node X_i to the node X_j, i.e. $(X_i, X_j) \in E$, if and only if the random variable X_i is a parent of the random variable X_j. Assuming that $X = \{X_1, X_2, ..., X_N\}$ is complete and closed system, directed edge (X_i, X_j) may be interpreted as "X_i causes X_j", while the lack of such an edge may be interpreted as "X_i does not influence X_j directly" or "X_i and X_j are conditionally independent".

Example

To better illustrate the idea of the probabilistic graphical models we will provide an example similar to the one from Pearl (2000). Lets consider the graphical model presented in Figure 1.5. This model consists of 5 variables, namely: the season (X_1, possible values are Spring, Summer, Autumn, Winter), whether sprinkler was on (X_2, yes/no), did it rain (X_3, yes/no), if the pavement is wet (X_4, yes/no) and finally if the pavement is slippery (X_5, yes/no). From the graph we can observe that "Season" directly influences how likely it is to "Rain", on the other hand we see that "Season" does not directly influence the probability of the pavement being "Wet" but indirectly via variables "Sprinkler" and "Rain", therefore there is no edge between "Season" and "Wet".

d-Separation

The concept of the d-separation was introduced in Pearl (2000). The d-separation (or directional-separation) is a criterion for deciding, from a given causal graph, whether a set of variables Y is independent of another set Z, given a third set W. The idea



$P(X_1 = i) = 1/4, i = 1, 2, 3, 4$		
$P(X_2 = 1) = 0.9$	$X_1 = 1$ or 2	
$P(X_2 = 1) = 0.2$	$X_1 = 3$ or 4	
$P(X_3 = 1) = 0.3$	$X_1 = 1$ or 2	
$P(X_3 = 1) = 0.8$	$X_1 = 3$ or 4	
$P(X_4 = 1) = 0.1$	$X_2 = 1$	$X_3 = 1$
$P(X_4 = 1) = 0.8$	$X_2 = 1$	$X_3 = 2$
$P(X_4 = 1) = 0.8$	$X_2 = 2$	$X_3 = 1$
$P(X_4 = 1) = 0.9$	$X_2 = 2$	$X_3 = 2$
$P(X_5 = 1) = 0.1$	$X_4 = 1$	
$P(X_4 = 1) = 0.8$	$X_4 = 2$	

Fig. 1.5 Example of a probabilistic graphical model.

is to associate the "dependence" with the "connectedness" (i.e., the existence of a connecting path) and the "independence" with the "disconnectedness" or "separation". The only adjustment to this simple idea is to rigorously define what we mean by a "connecting path", given that we are dealing with a system of directed arrows in which some vertices (those in the subset W) correspond to the measured variables, whose values are known precisely. To account for the orientations of the arrows we use the terms "d-separated" and "d-connected" (where d stands for "directional").
Consider three disjoint subsets of variables $Y, Z, W \subseteq X$, which are represented as subsets of nodes in a DAG G. We are interested in testing whether Y is independent of Z given W, in any distribution compatible with G, i.e. $Y \perp_G Z \mid W$. In order to do this we need to check whether the nodes corresponding to the variables in W block all the paths from the nodes in Y to the nodes in Z. A path can be either blocked, or open. Intuitively, 'to block' means to stop the flow of information (or correlation) between the variables connected by such paths. Whether a path is blocked or open can be determined using a criterion called the d-separation:

Definition 1.2.1. (d-separation Pearl (2000)): A path p is said to be *d-separated* (or blocked) by a set of nodes M iff

- p contains a chain $i \to m \to j$ or a fork $i \leftarrow m \to j$ (where i and j are not connected) such that the middle node m is in M, or
- p contains an inverted fork (or collider) $i \to m \leftarrow j$ such that the middle node m is not in M and such that no descendant of m is in M.

We say that a set of nodes I is d-separated from a set of nodes J given a set of nodes M iff each undirected path from every node $i \in I$ to every node $j \in J$ is d-separated by M. Finally we can define the conditional independence properties in DAG as $X_I \perp_G X_J \mid X_M$, written as d-sep$_G(Y; Z \mid W)$ if and only if I is d-separated from a set of nodes J given a set of nodes M (Murphy, 2012).
In order to understand the concept of the d-separation better lets consider the example from the Section 1.2.3. If we only know what is the "Season" (X_1), this provides us with some information about whether the pavement is "Wet" (X_4), i.e. X_1 and X_4 are not d-separated. On the other hand if we already know whether the "Sprinkler" (X_2) was on and whether there was "Rain" (X_3) last night, we have all the information about the probability of pavement being "Wet", that is knowing the "Season" does not provide us with any new information. In other words X_1 and X_4 are d-separated by the subset $\{X_2, X_3\}$, or d-sep$_G(X_1; X_4 \mid \{X_2, X_3\})$. This represents the first case of the Definition 1.2.1.
Now lets consider a different case. X_1 d-separates X_2 and X_3, i.e. if I know what "Season" it is now, then whether the "Sprinkler" will be on and whether it will "Rain" becomes independent random variables. On the other hand, if I also know whether the pavement is "Wet", this makes random variables "Sprinkler" and "Wet" dependent, for example if I know that the pavement is wet, but it did not rain last night, then it is very likely that the sprinkler must have been on. Therefore knowing X_4 opens the information flow between X_2 and X_3. This represents the second case of the Definition 1.2.1.

Bayesian network

Suppose we have a random variable $X = (X_1, ..., X_N)$. Then using the chain rule we can write the probability p of X as

$$p(X) = p(X_1, ..., X_n) = p(X_1)p(X_2 \mid X_1)p(X_3 \mid X_1, X_2)...p(X_N \mid X_1, ..., X_{N-1}) \tag{1.2.1}$$

even in the simplest case where all X_i are binary random variables, this will require $2^N - 1$ of parameters to define; 1 parameter for $p(X_1)$, 2 parameters for $p(X_2 \mid X_1)$ and up to 2^{N-1} parameters for $p(X_N \mid X_1, ..., X_{N-1})$. Now suppose the random variable X can be represented as a DAG. Then the number of parameters required can be significantly reduced. Given a random variable $X = (X_1, X_2, ..., X_N)$ and a DAG G, choose a topological ordering $X_{k_1}, X_{k_2}, ..., X_{k_N}$ of the elements of X, such that

$\text{Pa}_G(X_{k_i}) \in \{X_{k_1}, X_{k_2}, ..., X_{k_{i-1}}\}$. This can be done as G is a DAG. For brevity let us relabel X_{k_i} as X_i. Then for every X_i and any X_j, s.t. $X_j \notin \text{Pa}(X_i) \cup X_i$ and $j < i$, X_i and X_j are d-separated by $\text{Pa}(X_i)$, i.e. path $X_j \rightsquigarrow X_i$ is blocked by $\text{Pa}(X_i)$. And so, X_i and X_j are independent, given $\text{Pa}(X_i)$. Therefore $\Pr(X_i \mid X_1, ..., X_{i-1}) = \Pr(X_i \mid \text{Pa}(X_i))$. Combining the chain rule with the above, we get:

$$\begin{aligned} \Pr(X \mid G) &= \Pr(X_1, X_2, ..., X_n \mid G) \\ &= \prod_{i=1}^{n} \Pr(X_i \mid X_1, ..., X_{i-1}, G) \quad \text{by Chain Rule} \\ &= \prod_{i=1}^{n} \Pr(X_i \mid \text{Pa}_G(X_i)) \quad \text{by d-separation} \end{aligned} \tag{1.2.2}$$

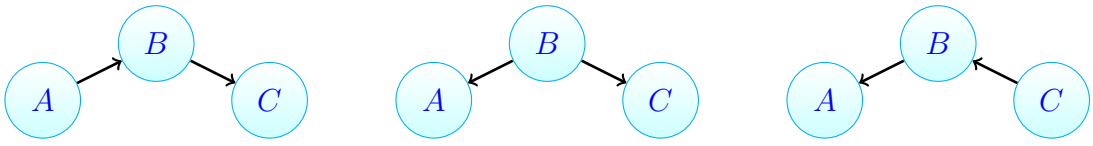
In the special case when the joint probability density of X can be factorised in this way, we call X a Bayesian network.

Definition 1.2.2. (Bayesian network): X is a Bayesian network with respect to a graph G if its joint probability density function (with respect to a product measure) can be written as a product of the individual density functions, conditional on their parent variables:

$$p(X) = \prod_{v \in V} p\left(X_v \mid \text{Pa}_G(X_v)\right) \tag{1.2.3}$$

Equivalent networks

Having a skeleton of a graph and all the d-separations in it, does not necessarily define the unique directed acyclic graph (DAG). For example see Figure 1.6. All three examples have the same skeleton, namely $A - B - C$, and same d-separation, namely $\text{d-sep}_G(A, C \mid B)$, but are different DAGs. Such networks are called equivalent.



(a) 1^{st} equivalent network. (b) 2^{nd} equivalent network. (c) 3^{rd} equivalent network.

Fig. 1.6 Examples of three graphs with the same skeleton and same d-separations.

Definition 1.2.3. (Equivalent network): Two networks are *equivalent* if they have same skeleton structures and the same v-structures (colliders). The set of all equivalent networks is called an *equivalence class*.

An equivalence class may be represented by a partially directed acyclic graph (PDAG). PDAG has both directed and undirected edges. Only the edges that have the same orientation in all the networks in the equivalence class will be directed in the PDAG. Edges that change their orientation (for example the edge $A - B$ in the example above can be oriented as $A \to B$ in 1.6a and as $A \leftarrow B$ in 1.6b) will be left undirected in the PDAG.

1.2.4 Conclusions

In this section we discussed probabilistic graphical models and how they can be used to put the knowledge about a real life system into a probabilistic framework. We also discussed ways of performing efficient computations in this set up, by using d-separation and factorising the joint probability density. In conclusion a PGM provides us with a probabilistic framework which is convenient to work with as well as easy to interpret; the existence or absence of the edges in the model may be directly interpreted as the existence or absence of direct influence between the variables.

1.3 PC Algorithm

The PC algorithm (named after its inventors Peter Spirtes and Clark Glymour (Spirtes and Glymour, 1990)) consists of three distinct phases. The first (skeleton) phase finds the skeleton of the PDAG, that is, it finds all adjacencies based on an independence test. The second (collider) phase finds all colliders (V_i, V_k, V_j) and directs edges $V_i \to V_k$ and $V_j \to V_k$. The third (transitive) phase applies the Meek rules (Meek, 1995) to extend all directions found in the collider phase to the rest of the PDAG. In this section we discuss each phase in more detail.

1.3.1 Skeleton Phase

In the skeleton phase we start from a fully connected graph $G = (V_G, E_G)$. Then for every pair of connected vertices $X, Y \in V_G$ (that is for each edge $\{X, Y\} \in E_G$) we test whether $X \perp Y \mid Z$, where Z is a subset of their neighbours (could be an empty set, then we would test for the unconditional independence). If $X \perp Y \mid Z$ for some Z we conclude that X does not directly influence Y (or the other way around) and delete the edge $\{X, Y\}$, we also memorize the subset Z as the separating set (or sepset(X, Y)) of X and Y. If $X \not\perp Y \mid Z$ for any of the subsets of the neighbours Z we keep the edge. The output of the skeleton phase is an undirected graph.

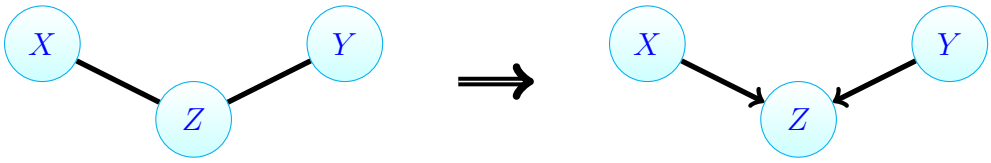
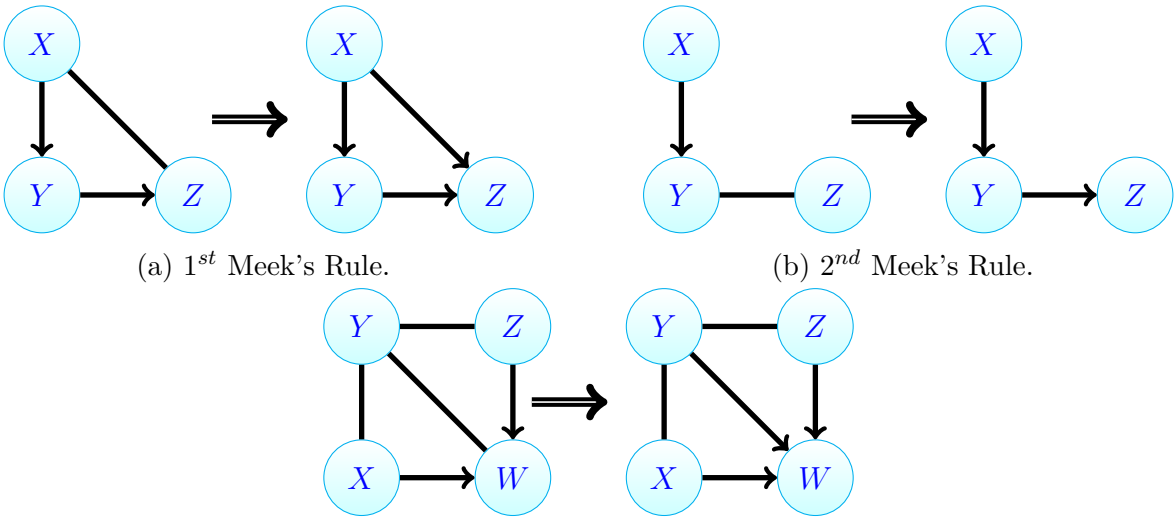


Fig. 1.7 Collider phase. $Z \notin \text{sepset}(X, Y)$.

If the neighbourhood of X and Y has size m, there are 2^m possible subsets Z that we have to test. In practice we usually only test for the conditional independence with the sets Z of size up to some k where $k < m$.
The PC algorithm can integrate a prior knowledge about the graph structure. Usually this prior knowledge comes in the form of "the edge $\{x, y\}$ is in G", i.e. we are not allowed to delete the edge $\{x, y\}$ or "the edge $\{z, w\}$ is not in G", i.e. we remove the edge $\{z, w\}$ without even testing. This approach is taken in the implementation of the PC algorithm in the `R` package `pcalg`. This form of adding prior knowledge to the algorithm does not work for a datasets like the Schistosomiasis one. In the Schistosomiasis dataset we know that variables belong to specified time slots: "before treatment", "24h after treatment", "9 weeks after treatment" and "8 months after treatment". In more general case suppose we have k non-overlapping time slots. When we test for the conditional independence between x and y which are from the time slots k_x and k_y respectively, then we need to consider only the neighbours of x and y from the time slots up to and including $\max(k_x, k_y)$. If a variable z is from a time slot later than $\max(k_x, k_y)$, it can only be a child of x and y. We know that x and y can be independent given z only if we are in one of the three cases: $x \to z \to y$ or $x \leftarrow z \leftarrow y$ or $x \leftarrow z \to y$. In all three cases z must be a parent of either x or y.

1.3.2 Collider Phase

In the collider phase we start directing edges of the graph G. For every unshielded collider (x, z, y), that is for every triple (x, z, y), where there are edges $\{x, z\}$ and $\{y, z\}$ but no edge between x and y, we check if the node z is in the separating set of x and y. If $z \in \text{sepset}(x, y)$, then we are in one of the three cases: $x \to z \to y$ or $x \leftarrow z \leftarrow y$ or $x \leftarrow z \to y$, therefore we do not orient any of the edges. If on the other hand $z \notin \text{sepset}(x, y)$ then it must be the case that $x \to z \leftarrow y$ and we orient the edges accordingly. The output of the collider phase is a partially directed graph containing both directed and undirected edges.



(a) 1^{st} Meek's Rule.

(b) 2^{nd} Meek's Rule.

(c) 3^{rd} Meek's Rule.

Fig. 1.8 Meek's rules.

1.3.3 Transitive Phase

In the transitive phase we repeatedly apply Meek's rules (shown in Figure 1.8). The first rule states, if we have directed edges $x \to y$ and $y \to z$ and an undirected edge $x - z$ we must orient it as $x \to z$, we do so because the opposite orientation would create a cycle, which is not allowed in the DAG. The second rule states, if we have a directed edge $x \to y$ and an undirected edge $y - z$ then we must direct it as $y \to z$, the opposite orientation would create an unshielded collider $x \to y \leftarrow z$ and we assume we have found all the colliders in the collider phase. The third rule is slightly more complicated: if we have $x \to w \leftarrow z$, $x - y - z$, there is no edge between x and z and there is an undirected edge $y - w$ we must direct it as $y \to w$. As $x - y - z$ is not directed as a collider, it must be the case that $y \in \text{sepset}(x, z)$ and therefore it is the case that either $y \to x$ or $y \to z$ (or both). In either case directing $y - w$ as $y \leftarrow w$ would create a cycle.
After repeatedly applying Meek's rules for as many times as possible we end up with a maximally directed PDAG.

1.4 Loss Function

In this section we introduce the concept of a *loss function*. In optimization, statistics and decision theory, the loss function (sometimes referred to as the cost function) is a function that maps a probabilistic event X and some rule δ onto a real number.

Intuitively a loss function can be interpreted as a cost incurred by choosing a rule δ when an event X occurs. The following definitions are taken from Wald (1950).

Definition 1.4.1. Loss Function: Let X be a random variable taking values in a sample space $(\chi, \mathcal{B}, P_\theta)$, $\theta \in \Theta$ and let $D = \{\delta\}$ be the space of all possible decisions that can be taken on the basis of an observed X. Then any lower bounded real-valued function $L : \Theta \times D \to \mathbb{R}$ is called a Loss Function. The value of a loss function L at an arbitrary point $(\theta, \delta) \in (\Theta, D)$ is interpreted as the cost incurred by taking a decision $\delta \in D$, when the true parameter is $\theta \in \Theta$.

Another important concept is the *expected loss*. It can be interpreted as the expected cost to be incurred if we follow the rule δ.

Definition 1.4.2. Expected Loss:

$$R(\theta, \delta) = \mathbb{E}_\theta\Big[l\Big(\theta, \delta(X)\Big)\Big] = \int_X l\Big(\theta, \delta(x)\Big) p_\theta(x) dx$$

Here, θ is a fixed but unknown state of nature, X is a vector of observations stochastically drawn from a population, E_θ is the expectation over all population values of X, p_θ is a probability measure over the event space of X (parametrized by θ) and the integral is evaluated over the entire support of X.

Examples of Loss Function

Here we provide some examples of possible loss functions.

- Square loss (L_2-norm)

  $$l(\theta, \delta(x)) = (\theta - \delta(x))^2$$

  Square loss works very well for the regression problems, though it may over penalize the outliers. It is minimized by the posterior mean.

- Absolute loss (L_1-norm)

  $$l(\theta, \delta(x)) = |\theta - \delta(x)|$$

  Absolute loss can be applied to the regression problems just as well as the square loss, also does not penalize the outliers as much. It is minimized by the the posterior median.

- ϵ-insensitive loss

  $$l(\theta, \delta(x)) = \max(0, |\theta - \delta(x)| - \epsilon)$$

  This loss function is ideal when small amounts of error are acceptable.

1.5 Kullback-Leibler Divergence

Given two random variables X and Y with the probability density functions p and q respectively we want to evaluate how similar these two pdfs are. For this task we require some measure of similarity between distributions. The most popular choice for such a measure is the Kullback-Leibler Divergence (D_{KL}). In this section we introduce the D_{KL} and some of its properties. Most of the theory is taken from Bishop (2006) and MacKay (2003).

Definition 1.5.1. Kullback-Leibler Divergence (discrete): For the discrete probability distributions $p(x)$ and $q(x)$, the Kullback-Leibler divergence is defined to be:

$$D_{\mathrm{KL}}(p\|q) = \sum_x p(x) \log \frac{p(x)}{q(x)}$$

Definition 1.5.2. Kullback-Leibler Divergence (continuous): For the continuous probability density functions p and q, the Kullback-Leibler divergence is defined to be:

$$D_{\mathrm{KL}}(p\|q) = \int_{-\infty}^{\infty} p(x) \log \frac{p(x)}{q(x)} dx$$

It is important to note that Kullback-Leibler divergence is not symmetric, i.e. $D_{\mathrm{KL}}(p\|q) \neq D_{\mathrm{KL}}(q\|p)$. Therefore D_{KL}, even though it is sometimes called "KL-distance", it is not strictly a distance.

Proposition 1.5.1. The Kullback-Leibler Divergence satisfies

$$D_{\mathrm{KL}} \geq 0$$

with equality if and only if $p(x) = q(x)$.

As shown in Proposition 1.5.1 D_{KL} is equal to zero if and only if distributions $p(x)$ and $q(x)$ are equal for all x we can use the Kullback-Leibler Divergence as a measure for similarity between two distributions.
Closely related to the Kullback-Leibler divergence is the concept of the *entropy*. Intuitively entropy is a measure of unpredictability.

Definition 1.5.3. For a random variable X with the probability density function p, the entropy H is defined as

$$H(X) = \mathrm{E}\big[-\log(p(X))] = \int p(x) \log p(x) dx$$

1.6 Datasets

In this section we introduce the datasets we are going to use throughout this thesis.

- Simulated datasets
- Experimental and simulated single-cell datasets
- Experimental Schistosomiasis dataset

The first set of datasets for testing all the algorithms will be simulated. In this case we know the correct underlying network, noise distribution and the functional forms of the relationships between the variables. The second set of datasets will be taken from Sachs et al. (2005) as well as datasets simulated from them (more detail in Section 1.6.2). These datasets allow us to have datasets with noise characteristics as close to the real noise characteristics as possible while still maintaining knowledge about the underlying networks. Finally the last set of datasets is from Schistosomiasis patients. In this case we do not know the underlying network. These datasets also have a substantial percentage of missing values.
It is plausible that all these datasets have non-linear relationships between variables as well as cyclic relationships.
It is important to point out that Sachs et al. (2005) and Schistosomiasis datasets were chosen as representative examples of experimental datasets rather than with a specific biology driven question in mind. This is because the aim of this thesis is model development and model comparison rather than answering a specific dataset dependant biological question. Other possible datasets to consider are ones used in Hyttinen et al. (2010). Datasets from Hyttinen et al. (2010) have less samples for the same number of variables as the datasets that we used. They compensate by having time series data, though models considered in this thesis are unable to use this additional information.

1.6.1 Simulated Datasets

In order to test any network inference algorithm, the first step is to check how well it copes with finding a network from simulated data. That is we choose a network G as well as its parameters θ and simulate data X from this network. Then we run the network inference algorithm on the data X to find a network $\hat{G}$. We expect that $\hat{G} = G$ or at least $\hat{G} \approx G$ using some metric. We are interested in how close to the correct network can we get.

In this thesis we explore three different types of algorithms and therefore need to be able to generate from three different network types:

- discrete data from Bayesian networks
- continuous data from Bayesian networks
- continuous data from cyclic networks

Chapter	data	relationships	cycles
1	discrete	non-linear	no
3	continuous	non-linear	no
4	continuous	non-linear	yes

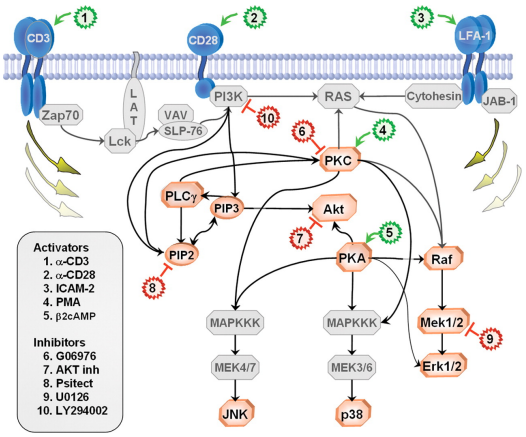
Table 1.1 Chapters and types of networks

1.6.2 Single-cell Datasets

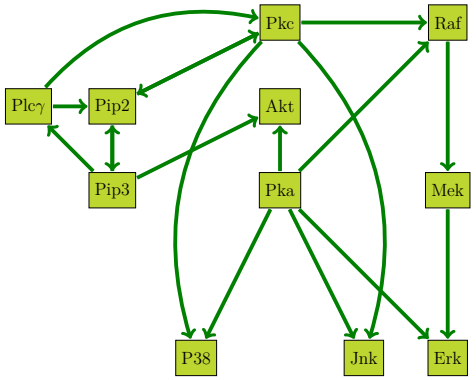
We are going to be using a well studied single-cell dataset on protein expression from Sachs et al. (2005). The study comprises eight experimental datasets of single cell measurements. Each dataset reports the expression level of eleven proteins: RAF, MEK, ERK (aka P44.42), PLCγ, PIP2, PIP3, PKC, AKT, PKA, JNK, P38. The number of observations (cells) varies from 700 to 900 cells per dataset. Each dataset is characterised by the quantitative value of protein expression response to a specific stimulatory cue or an inhibitory intervention (listed in Table 1.2).

Protein expression levels were obtained by flow cytometry measuring modification states of proteins, such as phosphorylation through antibodies. These are single cell measurements with each cell representing an independent observation. Only a few protein modifications are monitored. For example, PKC phosphorylates RAF at S497, S499, S259, however, only antibodies for RAF S259 were available. Consequently, some dependencies between protein states might be missed. Despite these shortcomings, some links between proteins are well established and shown in Figure 1.9a according to Sachs et al. (2005). The subnetwork of only the measured variables is shown in Figure 1.9b.

The graph in Figure 1.9b serves a twofold purpose. First, by exploiting the graph structure and by resampling the data, as described in more detail in Section 4.4.3, we generate datasets with characteristics close to real experimental datasets, but with

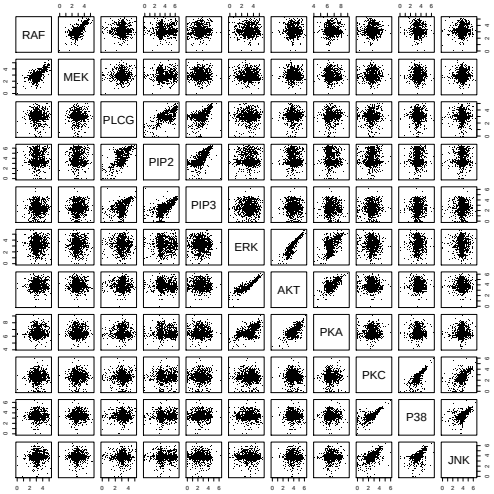


(a) Classic signaling network and points of intervention.

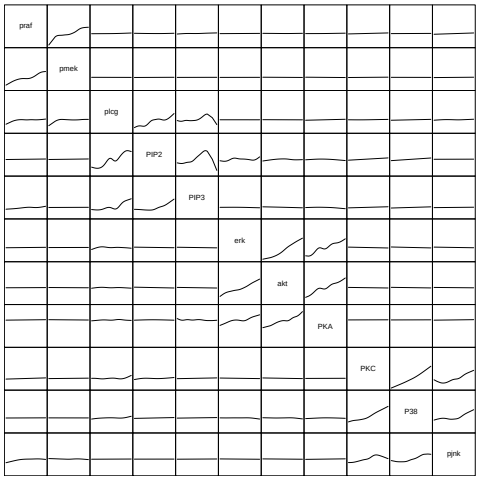


(b) Summary of known dependencies.

Fig. 1.9 Summary of known dependencies (after Sachs et al. (2005)).



(a) Scatterplots of the data in dataset 8.



(b) Regressions between proteins in the dataset 8. Cell in row i and column j should be interpreted as a functional relationship we get by regressing node i on node j.

Fig. 1.10 Experimental dataset 8 from Sachs et al. (2005) after log-transformation.

known dependencies. They will serve as test sets for comparing the performance of inference algorithms. Second, using the original datasets, the graph provides a reference network for measuring performance of algorithms on experimental data. Several limitations need to be kept in mind though. For some edges the direction of causal influence remains ambiguous. The inference algorithms considered in the

Dataset	Intervention type	Affected protein	Number of samples
1	General perturbation		853
2	General perturbation		902
3	Inhibition	AKT	911
4	Inhibition	PKC	723
5	Inhibition	PIP2	810
6	Inhibition	MEK	799
7	General perturbation		848
8	Activation	PKC	913

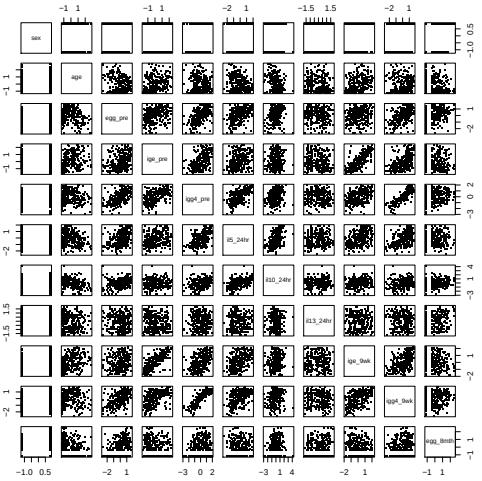
Table 1.2 Table of 8 single-cell datasets provided in Sachs et al. (2005).

first chapters (discrete Bayesian network MCMC and kernel PC) are based on the assumption that the dependency structure can be represented by a directed acyclic graph (we dropped this assumption in the last chapter where we use an MCMC sampler with a loss function). Such assumption is likely to hold only approximately true in real datasets. Also the selection of proteins is by no means complete and it is likely that latent, unobserved variables induce additional dependencies.
Figure 1.10 show dependencies (in the i'th row and j'th column is the signal function f of regression $X_j = f(X_i)$.) and pairwise scatterplots between variables of the dataset 8. Some dependencies are clear and close to linear, for example, RAF to MEK. Other dependencies are far less obvious, for example, RAF to PKC. This pattern of clear marginal dependencies between some of the related protein pairs but not all of them, is present in all eight datasets, as a reminder that network inference is not easily reducible to simple marginal correlation.

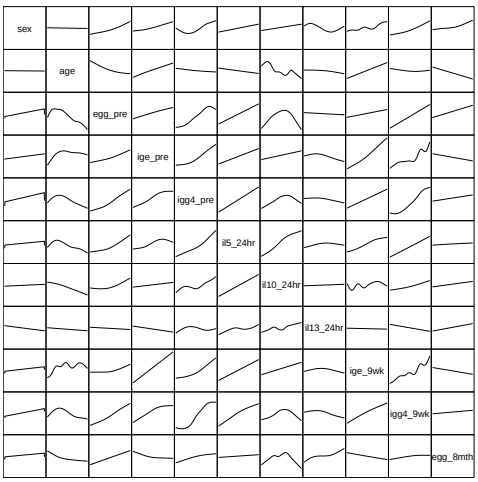
1.6.3 Schistosomiasis Dataset

For the Schistosomiasis dataset we do not have a reference network and this dataset has a large percentage of the data missing. Schistosomiasis is a waterborne parasitic infection caused by worms of the genus Schistosoma. The disease is endemic to over 70 countries in Africa, South America and Asia. Approximately 600 million people are at risk of infection, 200 million people are infected, and 200 000 deaths are associated with the disease every year (Thétiot-Laurent et al. (2013)). The majority of Schistosomiasis infections occur in children between the ages of 5 and 14 (The Carter Center (2014)) and the intensity of infection is higher in males than females (Tukahebwa

et al. (2013)). Schistosomiasis is the second most prevalent and socio-economically devastating parasitic disease in tropical countries after malaria.
The Schistosoma parasite goes through several stages in its lifespan. First, the eggs are eliminated from the human body with feces or urine. Then, the eggs hatch and release miracidia, which swim and penetrate into a specific type of fresh water snail in order to use it as an intermediate host. After leaving the snail, the infective larvae swim and penetrate the skin of a human host. They then live in blood vessels for an average of 7 years.
In its natural state the parasite lacks antigens, which makes it difficult for the immune system to recognise. After being treated with the drug Praziquantel the parasite dies, and its proteins trigger the immune system's response. As part of this response, cytokines are released; in particular interleukins such as IL-5, IL-10, IL-13, which in turn stimulate B-lymphocyte production. B-lymphocytes produce immunoglobulins. In this context, the immunoglobulin IgE is of particular interest (Dunne et al. (1992)). The key question in this study is how treatment with Praziquantel, which releases worm antigen into the blood, affects the levels of interleukins and immunoglobulins, particularly IgE, and how this response confers long-term immunity.
In this study, we use data from 450 individuals (225 men and 225 women, varying in age from 7 to 76), from a fishing community in Uganda, Lake Victoria. For each individual, 28 variables were measured. The variables include: demographic ones (sex, age), infection intensity (egg count in stool sample), levels of interleukins (Il-5, Il-10...), and the levels of immunoglobulins (IgG, IgE...). Measurements were taken before treatment, and then 24h, 9 weeks, 8 months and 2 years after the treatment.
From Figure 1.11b we observe that almost all variables have significant signal when regressing on each other. In contrast to the Sachs datasets where signal is observed in only few cliques. This will lead to a complicated conditional dependence structure. We also observe some highly non-linear relationships between variables. We also provide the scatterplots for the data in Figure 1.11a.
Missing data accounts for 17% of the total dataset. These values can be considered to be missing completely at random, as they are absent mainly due to problems related to the storage and analysis of blood and stool samples.
Based on previous work (Wernisch et al. (2007)) the dataset was reduced to 11 key variables for the initial model. The reduced set of variables is provided in the Table 1.3 and the full set of variables is shown in Appendix A, Table A.1.



(a) Scatterplots for the Schistosomiasis dataset.



(b) Signals in the Schistosomiasis dataset.

Fig. 1.11 Schistosomiasis dataset after the log-transformation

Name	Type	Time	Missing Values
Sex	Demographic	Before treatment	0
Age	Demographic	Before treatment	0
Egg count pre	Infection load	Before treatment	0
IgE pre	Immunoglobulin	Before treatment	24
IgG4 pre	Immunoglobulin	Before treatment	11
IL-5 24 hrs	Interleukin	24h after treatment	134
IL-10 24 hrs	Interleukin	24h after treatment	62
IL-13 24 hrs	Interleukin	24h after treatment	164
IgE 9 weeks	Immunoglogulin	9 weeks after treatment	29
IgG4 9 weeks	Immunoglogulin	9 weeks after treatment	26
Egg count 8 months	Infection load	8 months	28

Table 1.3 Schistosomiasis dataset; variables used in the initial model

Chapter 2

Network Inference of Discrete Bayesian Networks

2.1 Introduction

In this chapter we discuss network inference in the discrete Bayesian network setup. The aim of this chapter is to introduce the network structure search that we will use in Chapter 5. First we discuss the network inference problem in more detail. We provide an algorithm for MCMC sampling from the network space. Finally we discuss results, limitations and possible future extensions of the algorithm.

2.1.1 Network Inference Problem

Suppose that we have a causal network with N nodes, for example of the protein interactions in a cell (as in Sachs et al. (2005)) or an immune system response (as in Wernisch et al. (2007)). Also suppose that we have M samples from this network, this could be protein expression levels measured from M cells (as in Sachs et al. (2005)) or levels of immunoglobulins and interleukins measured from the blood samples of M patients (as in Wernisch et al. (2007)). Lets assume this network can be represented by some graph G. Our task is to find this graph G given the M data points we have sampled from the network.

Finding the correct graph G is a difficult combinatorial problem. Given that the size of the set of the nodes V_G is N, there are $2^{N(N-1)}$ possible graphs on N nodes: there are $N(N-1)$ possible edges (we allow both edges $X_i \to X_j$ and $X_i \leftarrow X_j$, for all $X_i, X_j \in V_G$) and each can be either "on" or "off". Even for a modest number of nodes the number of possible graphs is huge, for example if $N = 10$ there are

$2^{45} \approx 3.5 \times 10^{13}$ possible graphs. Exploring all the space is just not feasible. There are two options, either to apply some variation of a greedy search algorithm (for example a PC algorithm) or some stochastic search technique, for example, an MCMC sampler. A large search space is not the only issue that we may run into trying to find the graph G. In principal G may contain both directed and undirected edges. Let X and Y be two variables that we have observations of, represented by two nodes in G. We interpret a directed edge $X \to Y$ as "X directly influences Y". On the other hand an undirected edge $X - Y$ could be interpreted either as "X and Y are related, but neither X directly influences Y or vice versa" or "either X directly influences Y or Y directly influences X, but we cannot determine which one". The first case can happen due to a misspecified model, an example is presented in Figure 2.1. Z is not observed, we observe only the values of X and Y. The true structure is $X \leftarrow Z \to Y$. X and Y will be highly dependent on each other. On the other hand it is neither the case that $X \to Y$ nor $X \leftarrow Y$. Therefore we would expect to conclude that we will find only an undirected edge.

(a) True model. (b) Inferred model.

Fig. 2.1 Example of underlying true network that leads to undirected edge.

The second case can happen due to the symmetry in the model. Suppose data comes from $X \sim \mathcal{N}(0,1)$ and $Y \sim X + \mathcal{N}(0,1)$. Then both $X \to Y$ and $Y \to X$ would explain data equally well.

In the single-cell datasets from Sachs et al. (2005), we do not have observations of all the proteins in the network. Therefore we have to work with a subgraph shown in Figure 1.9b rather than the full network shown in Figure 1.9a. As shown in the simple example from Figure 2.1, having latent variables might hinder our chances of finding the true underlying network.

Another issue that may occur is missing values. For example in the Schistosomiasis dataset around 17% of all data is missing. As long as the data is missing at random, this should not cause insurmountable issues, but it would still reduce the ability of any algorithm to find the correct network.

2.1.2 Discrete Bayesian Network Inference

In this chapter we will introduce an approach to solve a special case of the problem proposed in Section 2.1.1, in particular an approach for the network inference of discrete Bayesian networks. Recall that a Bayesian network is a directed acyclic graph. This reduces the search space (the number of possible graphs) from $2^{N(N-1)}$ to less than $3^{\binom{N}{2}} = \sqrt{3}^{N(N-1)}$ ($\binom{N}{2}$ distinct pairs of nodes u and v which can be disconnected, connected $u \to v$ or $u \leftarrow v$). This space is still too large for exhaustive search. The real simplification comes from the fact that we are considering only acyclic graphs. This allows us to factorise the likelihood (this will be discussed in more detail in the Section 2.3). Another simplification that we are making is considering discrete data. Most of the time we will be presented with continuous data, it is natural to think about both the protein expressions as in (Sachs et al., 2005) or interleukin/immunoglobulin levels as in (Wernisch et al., 2007) as continuous variables. But discretizing the data reduces the computation time.

2.2 Structure Search

As we have discussed above, our problem requires to find a DAG G. The search space $\mathcal{G}$ in this case are all possible DAGs on N nodes. This space is too large to perform an exhaustive search. Performing any greedy search does not guarantee to find a global maximum - only a local one. In such a set up approximate sampling methods like Markov Chain Monte Carlo (or MCMC) are known to perform well (Tasaki et al., 2015). In this case we would define a probability for every possible graph on N nodes, i.e. let $p_i = p(G_i)$, for all $G_i \in \mathcal{G}$. We will use an MCMC sampling technique to sample the probability distribution p to get an empirical approximation. Given that we run the MCMC chain for long enough it is guaranteed to give an unbiased sample from the correct distribution.

It is worth noting that in this case our search space is discrete and MCMC search in discrete spaces is known to be notoriously hard. This is because there is no natural way of making the MCMC jump proposal smaller, as it would be the case in the continuous space (the distance between two points in $\mathcal{G}$ is at least one edge, i.e. it is bounded below).

Two most popular methods are the Gibbs sampler (Geman and Geman, 1984) and the Metropolis-Hastings method (Hastings, 1970; Metropolis et al., 1953). Suppose the current state is $G_i = \left(e_1, e_2, ..., e_{N(N-1)}\right) \in \{0,1\}^{N(N-1)}$, where $e_k \in \{0,1\}$, $k = 1, ..., N(N-1)$ defines the k^{th} edge of the graph: either 0 for no edge or 1 for an

edge. The Gibbs sampler draws a new value e_k from the full conditional distribution $\pi(e_k \mid e_{-k})$, where $e_{-k} = (e_j : j \neq k)$. In the network inference set up it would sample one edge at a time, given the rest of the structure. The Metropolis-Hastings algorithm proposes a jump from the state G_i to the state G_j with the probability $r(G_i, G_j)$ and accepts it with the probability

$$P(G_i \to G_j \mid X) = \min(1, A_{ij}) \tag{2.2.1}$$

where

$$A_{ij} = \frac{L(G_j \mid X)\pi(G_j)r(G_j, G_i)}{L(G_i \mid X)\pi(G_i)r(G_i, G_j)} \tag{2.2.2}$$

Here $L(G \mid X)$ is the likelihood of graph G given observed data X (can also be thought of as $P(X \mid G)$, i.e. the probability of observing data X if the true underlying model is G) and $\pi(G)$ is the prior of graph G.
In our search space, the natural way to transition between structures is to add an edge, to remove an edge or to revert an edge. These are the jump proposals in our MH algorithm. Suppose we are currently in state G, then we may propose to add a new edge $i \to j$, to remove an existing edge $i \to j$ or to revert an existing edge $i \to j$. Let K be the number of parents of node j, then these proposals have the following probabilities:

1. To add an edge: $r(G, G \cup \{i \to j\}) = \frac{1}{3N(N-K-1)}$
2. To remove an edge: $r(G, G \setminus \{i \to j\}) = \frac{1}{3NK}$
3. To revert an edge: $r(G, G \setminus \{i \to j\} \cup \{j \to i\}) = \frac{1}{3NK}$

Here we uniformly choose to add, remove or revert an edge; i.e. with probability $\frac{1}{3}$, we choose node j with probability $\frac{1}{N}$. Finally to add an edge we choose a non-parent node i with probability $\frac{1}{N-K-1}$, given that $K < N-1$, i.e. node j is not the child of all other nodes. To remove/revert an edge we choose a parent node i with probability $\frac{1}{K}$, given that $K > 0$, i.e. node j has at least one parent. If the condition $K < N-1$ for adding an edge or the condition $K > 0$ for removing/reverting an edge is not met, we cannot propose such a jump. We set the proposal ratio to zero and keep the current structure. If the required condition was met we get the following proposal ratios:

1. To add an edge: $\frac{r(G \cup \{i \to j\}, G)}{r(G, G \cup \{i \to j\})} = \frac{N-K-1}{K+1}$
2. To remove an edge: $\frac{r(G \setminus \{i \to j\}, G)}{r(G, G \setminus \{i \to j\})} = \frac{K}{N-K}$

3. To revert an edge: $\frac{r(G\setminus\{i\to j\}\cup\{j\to i\}),G)}{r(G,G\setminus\{i\to j\}\cup\{j\to i\})} = \frac{K}{K_{revert}+1}$

K_{revert} is the number of parents of node i (the original parent node). Note that the proposal ratios are consistent with the cases when the conditions were not met, i.e. they produce zero.
This covers the MCMC search algorithm for the structures. To be able to calculate the acceptance probability in Equation 2.2.2, we still need to find the likelihood of the graph G given data X, i.e. $L(G \mid X)\pi(G)$. We will discuss this in the next Section 2.3.

2.3 Likelihood

In this section we will discuss the likelihood for a discrete Bayesian network. We will also provide an efficient way of calculating it. First of all we should note that calculating

$$L(G \mid X) = \frac{p(X \mid G)\pi(G)}{\sum_{G_* \in \mathcal{G}} p(X \mid G_*)\pi(G_*)}$$

directly is not feasible, because finding the normalizing constant $\frac{1}{\sum_{G_*} p(X|G_*)\pi(G_*)}$ is computationally very expensive (we would need to calculate $p(X \mid G_*)$ for every G_* and, as noted above, the number of DAGs is superexponential in the number of nodes), but we can calculate $\frac{L(G_i|X)}{L(G_j|X)}$, as the normalizing constant cancels out. This is enough for us to be able to use the Metropolis-Hastings (MH) approach to sample from the space of DAGs.
We will use the fact that we are sampling only from directed acyclic graphs and therefore we can factorize the likelihood. Suppose we have a random variable $X = (X_1, ..., X_N)$. First of all using the chain rule we can write the probability p of X as

$$p(X) = p(X_1, ..., X_N) = p(X_1)p(X_2 \mid X_1)p(X_3 \mid X_1, X_2)...p(X_N \mid X_1, ..., X_{N-1}) \tag{2.3.1}$$

even in the simplest case where all X_i's are binary random variables, this will require a number of order 2^N of parameters to define. Now suppose the random variable X can be represented as a DAG. Then the number of parameters required can be significantly reduced. Given a random variable $X = \{X_1, X_2, ..., X_N\}$ and a DAG G, choose a topological ordering $X_{k_1}, X_{k_2}, ..., X_{k_N}$ of the elements of X, such that $\mathrm{Pa}_G(X_{k_i}) \in \{X_{k_1}, X_{k_2}, ..., X_{k_{i-1}}\}$. This can be done as G is a DAG. For brevity let us relabel X_{k_i} as X_i. Now for X_i and any X_j, s.t. $X_j \notin \mathrm{Pa}_G(X_i) \cup X_i$ and $j < i$ that is X_j is not a descendent of X_i, X_i and X_j are d-separated by $\mathrm{Pa}(X_i)$, i.e. path $X_j \rightsquigarrow X_i$

is blocked by $\text{Pa}_G(X_i)$. And so, X_i and X_j are independent, given $\text{Pa}_G(X_i)$. Therefore $\Pr(X_i \mid X_1, ..., X_{i-1}) = \Pr(X_i \mid \text{Pa}(X_i))$. Combining the chain rule with the above, we get:

$$\begin{aligned} p(X \mid G) &= p(X_1, X_2, ..., X_n \mid G) \\ &= \prod_{i=1}^{N} p(X_i \mid X_1, ..., X_{i-1}, G) \quad \text{by Chain Rule} \\ &= \prod_{i=1}^{N} p(X_i \mid \text{Pa}_G(X_i)) \quad \text{by d-separation} \end{aligned} \tag{2.3.2}$$

Note that when we are using the jump proposal from the set of {add an edge, remove and edge, revert an edge}, the parents of only one node will change.

- If we add an edge $\{i \to j\}$, only the parents of node j will change
- If we remove an edge $\{i \to j\}$, only the parents of node j will change
- If we revert an edge $\{i \to j\}$, only the parents of nodes i and j will change

Therefore in order to find the ratio of $\frac{p(X|G^*)}{p(X|G)}$ we will need to calculate only one term of the product from Equation 2.3.2.

Example

Suppose we proposed to add an edge $i \to j$. That is we are proposing a jump from the graph G to the graph $G^* = G \cup \{i \to j\}$. Then

$$\begin{aligned} \frac{L(G^* \mid X)}{L(G \mid X)} &= \frac{p(X \mid G^*)\pi(G^*)}{p(X \mid G)\pi(G)} = \frac{\prod_{i=1}^{N} p(X_i \mid \text{Pa}_{G^*}(X_i))}{\prod_{i=1}^{N} p(X_i \mid \text{Pa}_G(X_i))} \frac{\pi(G^*)}{\pi(G)} \\ &= \frac{p(X_j \mid \text{Pa}^*_G(X_j))}{p(X_j \mid \text{Pa}_G(X_j))} \frac{\pi(G^*)}{\pi(G)} \end{aligned} \tag{2.3.3}$$

As only the parents of the node X_j changed moving from G to G^* (that is $\text{Pa}_{G^*}(X_j) = \text{Pa}_{G^*}(X_j) \cup X_i$ and $\text{Pa}_{G^*}(X_k) = \text{Pa}_G(X_k)$, for all $k \neq j$), we need to calculate only one term of the new likelihood, which significantly speeds up the process.

2.3.1 Likelihood for Discrete Variable

In this section we provide full details for the likelihood for discrete variables as the introduction to Proposition 2.3.2. We are considering a network with N nodes and we have M observations. We start with one node X_v (one random variable). X_v

is an M-dimensional discrete vector with L_v possible values for each component, i.e. $X_v = (X_{v1}, ..., X_{vM}) \in \{1, ..., L_v\}^M$. We define all the possible parent nodes configurations as $C_v = (C_{v1}, ..., C_{vQ_v}) \in \{1, ..., L_v\}^{Q_v}$, where $\mathrm{Q}_v = \prod_{u \in \mathrm{Pa}_{G(X_v)}} L_u$ and L_u is the number of possible values the node X_u can take. This node is summarised by a $Q_v \times L_v$-dimensional vector $n_v = (n_{vql} : q = 1, ..., Q_v, l = 1, ..., L_v)$, where

$$n_{vql} = \sum_{i=1}^{M} \mathbf{1}\left(X_{vi} = l\right) \mathbf{1}\left(\mathrm{Pa}_G(X_{vi}) = C_{vq}\right), \forall l = 1, ..., L_v \tag{2.3.4}$$

here L_v is the number of distinct values the random variable X_v can take, Q_v is the number of possible parent nodes configurations and $i = 1, ..., M$ run through all M observations. So n_{vql} is the number of times the variable X_v had value l while having the parent nodes configuration q.
We assume that the n_v is distributed according to a Multinomial distribution with parameter

$$\theta = (\theta_{ql} : q = 1, ..., Q_v; l = 1, ..., L_v)$$

and a Dirichlet prior

$$\alpha = (\alpha_{ql} : q = 1, ..., Q_v; l = 1, ..., L_v)$$

That is

$$\begin{aligned} p(n_v \mid \theta) &= \prod_{q=1}^{Q_v} \prod_{l=1}^{L_v} \theta_{vql}^{n_{vql}} \\ p(\theta \mid \alpha) &= \frac{1}{Z} \prod_{q=1}^{Q_v} \prod_{l=1}^{L_v} \theta_{vql}^{\alpha_{vql}} \\ Z &= \int \prod_{q=1}^{Q_v} \prod_{l=1}^{L_v} \theta_{vql}^{\alpha_{vql}} d\theta = \prod_{q=1}^{Q_v} \frac{\prod_{l=1}^{L_v} \Gamma(\alpha_{vql})}{\Gamma(\sum_{l=1}^{L_v} \alpha_{vql})} \end{aligned} \tag{2.3.5}$$

Given this set up, we integrate out the parameter for a Multinomial distribution θ, to obtain a conditional probability distribution $p(n \mid \alpha)$:

$$
\begin{aligned}
p(n_v \mid \alpha) &= \int p(n \mid \theta) p(\theta \mid \alpha) d\theta \\
&= \prod_{q=1}^{Q_v} \frac{\Gamma(\sum_{l=1}^{L_v} \alpha_{vql})}{\prod_{l=1}^{L_v} \Gamma(\alpha_{vql})} \int \prod_{q=1}^{Q_v} \prod_{l=1}^{L_v} \theta_{vql}^{n_{vql}+\alpha_{vql}-1} d\theta \\
&= \prod_{q=1}^{Q_v} \frac{\Gamma(\sum_{l=1}^{L_v} \alpha_{vql})}{\prod_{l=1}^{L_v} \Gamma(\alpha_{vql})} \prod_{q=1}^{Q_v} \frac{\prod_{l=1}^{L_v} \Gamma(n_{vql} + \alpha_{vql})}{\Gamma(\sum_{l=1}^{L_v} n_{vql} + \alpha_{vql})} \\
&= \prod_{q=1}^{Q_v} \frac{\Gamma(\sum_{l=1}^{L_v} \alpha_{vql})}{\Gamma(\sum_{l=1}^{L_v} n_{vql} + \alpha_{vql})} \prod_{l=1}^{L_v} \frac{\Gamma(n_{vql} + \alpha_{vql})}{\Gamma(\alpha_{vql})}
\end{aligned}
\tag{2.3.6}
$$

Now we can write down the explicit form of the probability to observe data X given the graph structure G and a prior α (equivalently the likelihood of the graph structure G given data X). This is quite a convoluted definition where each parameter requires many subscripts, so we will provide an explanatory example at the end of the subsection. From Equation 2.3.2 it follows that the likelihood of a network is just a product of the likelihoods of each node, therefore:

$$
p(n \mid G, \alpha) = \prod_{v=1}^{N} \prod_{j=1}^{Q_v} \frac{\Gamma(\sum_{i=1}^{L_v} \alpha_{vql})}{\Gamma(\sum_{l=1}^{L_v} n_{vql} + \alpha_{vql})} \prod_{l=1}^{L_v} \frac{\Gamma(n_{vql} + \alpha_{vql})}{\Gamma(\alpha_{vql})} \tag{2.3.7}
$$

And so:

$$
l(G \mid n, \alpha) = \log p(n \mid G, \alpha) = \sum_{v=1}^{N} \sum_{j=1}^{Q_v} \left(\frac{\Gamma(\sum_{l=1}^{L_v} \alpha_{vql})}{\Gamma(\sum_{l=1}^{L_v} n_{vql} + \alpha_{vql})} + \sum_{l=1}^{L_v} \frac{\Gamma(n_{vql} + \alpha_{vql})}{\Gamma(\alpha_{vql})} \right) \tag{2.3.8}
$$

The likelihood in Equation. 2.3.8 is a sum. So after proposing an MH jump (to add, remove or revert an edge) $G_n \to G_*$, when we need to compare $l(G_n \mid X)$ versus $l(G_* \mid X)$ we need to calculate only a few terms of the new likelihood rather than all of it.

Example

Lets consider a very simple example as shown in Figure 2.2. Let all three random variables X, Y and Z be binary, i.e. $X, Y, Z \in \{1, 2\}$ and follow the probability distribution as defined in the Table 2.1 .

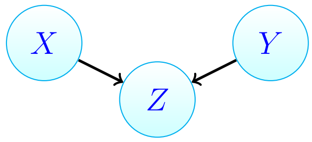


Fig. 2.2 Graph G of the Bayesian Network.

$P(X=1)=0.5$
$P(Y=1)=0.5$
$P(Z=1 \mid X=1, Y=1)=0.9$
$P(Z=1 \mid X=1, Y=2)=0.8$
$P(Z=1 \mid X=2, Y=1)=0.8$
$P(Z=1 \mid X=2, Y=2)=0.1$

Table 2.1 Probabilities of the Bayesian Network.

$n_{X1}=528$	$n_{X2}=472$
$n_{Y1}=499$	$n_{Y2}=501$
$n_{Z11}=252$	$n_{Z12}=21$
$n_{Z21}=211$	$n_{Z22}=44$
$n_{Z31}=172$	$n_{Z32}=54$
$n_{Z41}=26$	$n_{Z42}=220$

Table 2.2 Data generated from the Bayesian Network.

Also suppose we have a 1000 observations. They can be summarised as in Table 2.2. The q in the n_{Zq0} stands for the parents configuration: that is $q=1$ means $X=Y=0$, $q=2$ means $X=0$ and $Y=1$, $q=3$ means $X=1$ and $Y=0$ and finally $q=4$ means $X=Y=1$. As X and Y have no parents, they have only one "parents configuration" (could think about it as an empty set), while Z has 2 parents and each parent can attain 2 values, so it in total has $2^2=4$ parents configurations. Finally we have a uniform prior $\alpha_{vql}=\alpha=1$, for all v, q, l. The likelihood of G given data is

$$\begin{aligned} p(n \mid G,\alpha) &= p(n_X \mid G,\alpha)p(n_Y \mid G,\alpha)p(n_Z \mid G,\alpha) \\ &= \underbrace{\frac{\Gamma(2)}{\Gamma(529+473)}\frac{\Gamma(529)}{\Gamma(1)}\frac{\Gamma(473)}{\Gamma(1)}}_{n_X} \times \underbrace{\frac{\Gamma(2)}{\Gamma(500+502)}\frac{\Gamma(500)}{\Gamma(1)}\frac{\Gamma(502)}{\Gamma(1)}}_{n_Y} \\ &\quad \times \underbrace{\frac{\Gamma(2)}{\Gamma(253+22)}\frac{\Gamma(253)}{\Gamma(1)}\frac{\Gamma(22)}{\Gamma(1)}}_{n_Z \mid X=1,\, Y=1} \times \underbrace{\frac{\Gamma(2)}{\Gamma(212+45)}\frac{\Gamma(212)}{\Gamma(1)}\frac{\Gamma(45)}{\Gamma(1)}}_{n_Z \mid X=1,\, Y=2} \\ &\quad \times \underbrace{\frac{\Gamma(2)}{\Gamma(173+55)}\frac{\Gamma(173)}{\Gamma(1)}\frac{\Gamma(55)}{\Gamma(1)}}_{n_Z \mid X=2,\, Y=1} \times \underbrace{\frac{\Gamma(2)}{\Gamma(27+221)}\frac{\Gamma(27)}{\Gamma(1)}\frac{\Gamma(221)}{\Gamma(1)}}_{n_Z \mid X=2,\, Y=2} \end{aligned} \tag{2.3.9}$$

2.3.2 Priors

There are two options to incorporate a prior knowledge in the network inference. We can either use a prior on the structure or on the parameters.
There are couple of ways we can put a prior on the structure. We may put a prior directly on the edges: for example in the Schistosomiasis dataset we have prior information about when each variable was measured ("before treatment", "24h after", "9 weeks after", "8 months after" or "2 years after"). That means we have a partial ordering of the nodes of the graph, a variable can only be influenced by other variables from the same or earlier time slot. Therefore, we only allow edges $\{v_i \to v_j\}$, such that v_i belongs to the same or previous time slice as v_j. In the case of the datasets from Sachs et al. (2005) we have the reference network, so we may encourage the edges that are known to be present in the network. This considered, we might be more interested in using a non-informative prior in order to test the efficiency of an algorithm. An example of a non-informative prior would be a uniform prior, where every edge has the same prior probability of being in the network. Another approach could be to put a prior on an overall number of edges in the network. For example if we a priori believe the network to have approximately $q\%$ of all possible edges we may use a binomial prior:

$$\pi(G) = \binom{E}{e} q^e (1-q)^{E-e} \tag{2.3.10}$$

where $e = |E_G|$ is the number of edges in G and E is the total number of possible edges (if G has N nodes, then $E = \binom{N}{2}$).
Our other option is to put priors on the parameters. In the discrete Bayesian network set up, that would be the prior α for the conditional probability densities (or CPDs) θ in the Multinomial distribution. In our model we use Dirichlet priors since they are conjugate to the Multinomial distribution (as shown in Equation 2.3.5). The prior parameter α can be thought of as the fictitious observations that we will add to the real ones.

2.3.3 Likelihood Equivalence

In the setup, as previously defined, the likelihood equivalence preserving priors are often used on the CPDs. These priors are such that the likelihood of the model $(X_1 \to X_2)$ is equal to the likelihood of the model $(X_1 \leftarrow X_2)$. The idea behind this is that from purely observational data we only infer conditional independence properties and not

causality (for example MacKay (2003)). This is not necessarily true for all types of graphical models. We will discuss some ways to break the equivalence between these models in the later chapters. But in the case of discrete Bayesian networks (or for example linear models with Gaussian noise) it is indeed impossible to determine the direction of causality from observed data.
An example of the likelihood equivalence preserving prior would be

$$\alpha_{vql} = \frac{\alpha}{L_v \times \mathrm{Q}_v}, \text{ for all } q = 1, ..., \mathrm{Q}_v;\ l = 1, ..., L_v \tag{2.3.11}$$

recall that Q_v is the number of parent configurations for the node X_v, L_v is the number of distinct values the node X_v can attain and α is just a scaling constant.

2.3.4 Likelihood Equivalence Non-Preserving Priors

Despite the fact that likelihood equivalence preserving priors are often chosen, in some special cases we might be interested in breaking the symmetry they provide. Now we will state and prove some results about likelihood equivalence non-preserving priors. We assume a simple network of two nodes with two levels each

$$\begin{aligned} l_1 &= l(X_1 \to X_2 \mid n, \alpha) \\ l_2 &= l(X_1 \leftarrow X_2 \mid n, \alpha) \\ n &= (n_{11}, n_{12}, n_{21}, n_{22}) \\ \alpha &= (\alpha_{11}, \alpha_{12}, \alpha_{21}, \alpha_{22}, \alpha_{1*}, \alpha_{2*}, \alpha_{*1}, \alpha_{*2}) \end{aligned} \tag{2.3.12}$$

The Dirichlet prior α_{ij} is the number of the fictitious observations in which $X_1 = i$ and $X_2 = j$. α_{i*} is the number of the fictitious observations in which $X_1 = i$, and similarly, α_{*j} is the number of the fictitious observations in which $X_2 = j$. For the likelihood preserving prior $\alpha_{i*} = \alpha_{i1} + \alpha_{i2}$, i.e. the number of fictitious observations where $(X_1, X_2) = (i, 1)$ and $(X_1, X_2) = (i, 2)$ adds up to the number of fictitious observations where $X_1 = i$. By deviating from this prior we can control whether a variable with more uncertainty will be more or less likely to be the causal one. For example in the case of a uniform prior $\alpha_{i*} = \alpha_{ij} = \alpha$, the variable with less uncertainty is more likely to be causal.

First we prove a lemma about the difference between two log gamma functions.

Lemma 2.3.1. For any real numbers x and ϵ, if $\epsilon \ll x$, then

$$\log \Gamma(x + \epsilon) - \log \Gamma(x) \approx \epsilon \log(x)$$

Proof. We use an approximation for the gamma function from Whittaker and Watson (1996) p. 261

$$\log\Gamma(x) = (x - \frac{1}{2})\log(x) - x + \log(\sqrt{2\pi}) + O(x^{-1}) \tag{2.3.13}$$

it follows that

$$\begin{aligned}
\log\Gamma(x+\epsilon) - \log\Gamma(x) &= (x+\epsilon-\frac{1}{2})\log(x+\epsilon) - x - \epsilon + \log(\sqrt{2\pi}) \\
&\quad - (x-\frac{1}{2})\log(x) + x - \log(\sqrt{2\pi}) + O(x^{-1}) \\
&= \left(x-\frac{1}{2}\right)\log\left(\frac{x+\epsilon}{x}\right) + \epsilon\log\left(x+\epsilon\right) - \epsilon + O(x^{-1}) \\
&= \left(x-\frac{1}{2}\right)\log\left(1+\frac{\epsilon}{x}\right) + \epsilon\left(\log\left(x\right) + \log\left(1+\frac{\epsilon}{x}\right)\right) - \epsilon + O(x^{-1}) \\
&= \left(x-\frac{1}{2}\right)\left(\frac{\epsilon}{x} + O(x^{-2})\right) + \epsilon\left(\log(x) + \left(\frac{\epsilon}{x} + O(x^{-2})\right)\right) - \epsilon + O(x^{-1}) \\
&= \epsilon\log(x) + O(x^{-1})
\end{aligned} \tag{2.3.14}$$

where penultimate equation holds due to the Taylor approximation for $\log(1+\frac{\epsilon}{x}) = \frac{\epsilon}{x} + O(x^{-2})$ as $\epsilon \ll x$.

□

Proposition 2.3.2. For a prior uniform for the nodes with the same number of parents ($\alpha_{ij} = \alpha_1$, and $\alpha_{i*} = \alpha_{*j} = \alpha_0$, for all $i, j = 1, 2$), then $\Delta l = l_1$ - l_2 is proportional to $(2\alpha_1 - \alpha_0)$.

Proof. The log likelihood for the general case is given by

$$\log p(n \mid G, \alpha) = \sum_{v=1}^{N}\sum_{q=1}^{Q_v}\left(\frac{\Gamma(\sum_{l=1}^{L_v}\alpha_{vql})}{\Gamma(\sum_{l=1}^{L_v} n_{vql} + \alpha_{vql})} + \sum_{l=1}^{L_v}\frac{\Gamma(n_{vql}+\alpha_{vql})}{\Gamma(\alpha_{vql})}\right) \tag{2.3.15}$$

Recall that the suffix v stands for the node, q for parents configuration and l for the level. For model $G_1 = (X_1 \to X_2)$ we can simplify the notation. X_1 has no parents, so $n_{1ql} = n_{11l} = n_{l1} + n_{l2}$ and as X_2 has one parent X_1 which can attain two values, 1 and 2 we have $n_{2ql} = n_{ql}$. Similarly for $\alpha_{11l} = \alpha_0$, for $l = 1, 2$ and $\alpha_{2ql} = \alpha_1$, for $q, l = 1, 2$, here the subscript n for α_n represents the number of parents. Equation 2.3.15 simplifies to

$$\begin{aligned}
\log p(n \mid G_1, \alpha) = &+ \log\Gamma(\alpha_0 + \alpha_0) - \log\Gamma(\alpha_0) - \log\Gamma(\alpha_0) \\
&- \log\Gamma(n_{11} + n_{12} + n_{21} + n_{22} + \alpha_0 + \alpha_0) \\
&\underbrace{+ \log\Gamma(n_{11} + n_{12} + \alpha_0) + \log\Gamma(n_{21} + n_{22} + \alpha_0)}_{\text{terms for } X_1} \\
&+ \log\Gamma(\alpha_1 + \alpha_1) - \log\Gamma(\alpha_1) - \log\Gamma(\alpha_1) \\
&\underbrace{- \log\Gamma(n_{11} + n_{12} + \alpha_1 + \alpha_1) + \log\Gamma(n_{11} + \alpha_1) + \log\Gamma(n_{12} + \alpha_1)}_{\text{terms for } X_2 \mid X_1 = 1} \\
&+ \log\Gamma(\alpha_1 + \alpha_1) - \log\Gamma(\alpha_1) - \log\Gamma(\alpha_1) \\
&\underbrace{- \log\Gamma(n_{21} + n_{22} + \alpha_1 + \alpha_1) + \log\Gamma(n_{21} + \alpha_1) + \log\Gamma(n_{22} + \alpha_2)}_{\text{terms for } X_2 \mid X_1 = 2}
\end{aligned} \tag{2.3.16}$$

We get a similar expression for the log likelihood of model $G_2 = X_1 \leftarrow X_2$. We define $\Delta l = l_1 - l_2$ and we can write it as

$$\begin{aligned}
\Delta l = &+ \log\Gamma(n_{11} + n_{12} + \alpha_0) - \log\Gamma(n_{11} + n_{12} + \alpha_1 + \alpha_1) \\
&+ \log\Gamma(n_{21} + n_{22} + \alpha_0) - \log\Gamma(n_{21} + n_{22} + \alpha_1 + \alpha_1) \\
&- \log\Gamma(n_{11} + n_{21} + \alpha_0) + \log\Gamma(n_{11} + n_{21} + \alpha_1 + \alpha_1) \\
&- \log\Gamma(n_{12} + n_{22} + \alpha_0) + \log\Gamma(n_{12} + n_{22} + \alpha_1 + \alpha_1)
\end{aligned} \tag{2.3.17}$$

If $\alpha_0 = 2\alpha_1$ then $\Delta l = 0$. This represents the likelihood equivalence preserving priors. Suppose $\alpha_0 \neq 2\alpha_1$, then we can we define ϵ as $\epsilon = \alpha_0 - 2\alpha_1$ and we use Lemma 2.3.1 to find the relationship of Δl and ϵ.
Δl is a sum of terms of the form

$$\log\Gamma(n_{ij} + n_{kl} + \alpha_0) - \log\Gamma(n_{ij} + n_{kl} + \alpha_1 + \alpha_1)$$

for $i, j = 1, 2$. As the prior α is much smaller than the number of observations n we may apply Lemma 2.3.1 to get

$$\begin{aligned}
&\log\Gamma(n_{ij} + n_{kl} + \alpha_0) - \log\Gamma(n_{ij} + n_{kl} + 2\alpha_1) \\
&= \log\Gamma(n_{ij} + n_{kl} + 2\alpha_1 + \epsilon) - \log\Gamma(n_{ij} + n_{kl} + 2\alpha_1) \\
&\approx \epsilon \log(n_{ij} + n_{kl} + 2\alpha_1)
\end{aligned} \tag{2.3.18}$$

using notation $n_{i*} = n_{i1} + n_{i2} + 2\alpha_1$ and similarly for n_{*j} with $i, j = 1, 2$ we simplify Equation 2.3.17 to

$$\Delta l \approx \epsilon \left(\log(n_{1*}) + \log(n_{2*}) - \log(n_{*1}) - \log(n_{*2})\right) \tag{2.3.19}$$

$(\log(n_{1*}) + \log(n_{2*}) - \log(n_{*1}) - \log(n_{*2}))$ is constant and this proves the claim. □

Corollary 2.3.1. For a uniform prior ($\alpha_{ij} = \alpha_{i*} = \alpha_{*j} = \alpha$, for all $i, j = 1, 2$), the variable with a smaller variance is more likely to be the cause, i.e. $l_1 > l_2$ iff $\mathrm{Var}(X_1) \leq \mathrm{Var}(X_2)$.

Proof. In Proposition 2.3.2 we showed that

$$\Delta l \approx \epsilon \left(\log(n_{1*}) + \log(n_{2*}) - \log(n_{*1}) - \log(n_{*2})\right) \tag{2.3.20}$$

where $n_{i*} = n_{i1} + n_{i2} + 2\alpha$ and in this case $\epsilon = \alpha - 2\alpha = -\alpha$. By adding and subtracting two $\alpha \log(n)$ where $n = 4\alpha + \sum_{i,j} n_{ij}$, we get

$$\begin{aligned}
\Delta l &\approx -\alpha \left(\log(n_{1*}) + \log(n_{2*}) - 2\log(n) - \log(n_{*1}) - \log(n_{*2}) + 2\log(n)\right) \\
&\approx -\alpha \left(\log(p_{11}) + \log(p_{12}) - \log(p_{21}) - \log(p_{22})\right) \\
&= -\alpha \left(\log(p_{11}p_{12}) - \log(p_{21}p_{22})\right) \\
&= -\alpha \left(\log(\mathrm{Var}(X_1)) - \log(\mathrm{Var}(X_2))\right) \\
&= \alpha \left(\log(\mathrm{Var}(X_2)) - \log(\mathrm{Var}(X_1))\right)
\end{aligned} \tag{2.3.21}$$

here $p_{1j} = \frac{n_{j*}}{n} = p(X_1 = j)$, similarly for p_{2j}. Both variables are Bernoulli random variables, so $\mathrm{Var}(X_i) = p_{i1}p_{i2}$. We conclude that if we use the uniform prior, then the variable with a smaller variance is more likely to be the cause, i.e. $\mathrm{Var}(X_2) > \mathrm{Var}(X_1) \Rightarrow \log(\mathrm{Var}(X_2)) > \log(\mathrm{Var}(X_1)) \Rightarrow \Delta l > 0 \Rightarrow l_1 > l_2 \Rightarrow p(X_1 \to X_2 \mid n, \alpha) > p(X_1 \leftarrow X_2 \mid n, \alpha)$.

□

This result naturally extends to random variables with more than two outcomes. We are considering network with two nodes with L levels each

$$\begin{aligned}
l_1 &= l(X_1 \to X_2 \mid n, \alpha) \\
l_2 &= l(X_1 \leftarrow X_2 \mid n, \alpha) \\
n &= (n_{ij};\ i, j = 1, ..., L) \\
\alpha &= (\alpha_{ij}, \alpha_{i*}, \alpha_{*j};\ i, j = 1, ..., L)
\end{aligned} \tag{2.3.22}$$

Proposition 2.3.3. For a prior uniform for the nodes with the same number of parents ($\alpha_{ij} = \alpha_1$, and $\alpha_{i*} = \alpha_{*j} = \alpha_0$, for all $i, j = 1, 2$), then $\Delta l = l_1$ - l_2 is proportional to $(L\alpha_1 - \alpha_0)$.

Proof. Proof goes exactly same way as the proof for Proposition 2.3.2, except in the very last step we get

$$\Delta l \approx \epsilon \left(\sum_{l=1}^{L} \log(n_{l*}) - \sum_{l=1}^{L} \log(n_{*l}) \right) \tag{2.3.23}$$

where similarly as before $n_{i*} = \sum_{l=1}^{L} n_{il} + L\alpha_1$

□

Corollary 2.3.2. Let $p_{1l} = \frac{n_{l*}}{n} = p(X_1 = l)$ and $p_{2l} = \frac{n_{*l}}{n} = p(X_2 = l)$ for $l = 1, ..., L$. Then for a uniform prior ($\alpha_{ij} = \alpha_{i*} = \alpha_{*j} = \alpha$, for all $i, j = 1, 2$), the variable with a smaller product of probabilities is more likely to be the cause, i.e. $l_1 > l_2$ iff $\prod_j p_{1j} \leq \prod_j p_{2j}$.

Proof. As in the proof of the previous corollary we start by adding and subtracting $L \log(n)$ to the expression of Δl

$$\begin{aligned}
\Delta l &\approx (1 - L)\alpha \left(\sum_{l=1}^{L} \log(n_{l*}) - L\log(n) - \sum_{l=1}^{L} \log(n_{*l}) + L\log(n) \right) \\
&= (1 - L)\alpha \left(\sum_{l=1}^{L} \log\left(\frac{n_{l*}}{n}\right) - \sum_{l=1}^{L} \log\left(\frac{n_{*l}}{n}\right) \right) \\
&= (1 - L)\alpha \left(\sum_{l=1}^{L} \log\left(p_{1l}\right) - \sum_{l=1}^{L} \log\left(p_{2l}\right) \right) \\
&= (1 - L)\alpha \left(\log\left(\prod_{l=1}^{L} p_{1l}\right) - \log\left(\prod_{l=1}^{L} p_{2l}\right) \right)
\end{aligned} \tag{2.3.24}$$

This proves the claim.

□

It is difficult to interpret the meaning of $\prod_j p_{1j}$. We would suggest to think about it as a Kullback-Leibler divergence between uniform random variable X_0 and X_1.
Let X_0 be a uniform discrete random variable with L levels with probability density function p_0, i.e. $p_0(X_0 = l) = 1/L$ for all $l = 1, ..., L$. Then the Kullback-Leibler divergence (for more detail see Section 1.5) between X_0 and X_1 is defined as

$$D_{KL}(p_0 \| p_1) = \sum_{l=1}^{L} \frac{1}{L} \log\left(\frac{p_{1l}}{1/L}\right) \tag{2.3.25}$$

we can simplify it to

$$D_{KL}(p_0\|p_1) = \frac{1}{L}\sum_{l=1}^{L}\log(p_{1l}) + \frac{1}{L}\sum_{l=1}^{L}\log(L) \tag{2.3.26}$$

The difference between the products of probabilities for X_1 and X_2 can be written in terms of the Kullback-Leibler divergences

$$\left(\sum_{l=1}^{L}\log(p_{1l}) - \sum_{l=1}^{L}\log(p_{2l})\right) = L\left(D_{KL}(p_0\|p_1) - D_{KL}(p_0\|p_2)\right) \tag{2.3.27}$$

We can rephrase Corollary 2.3.2 as: the random variable with smaller Kullback-Leibler divergence from a uniform distribution is more likely to be the cause.

2.4 Missing Value Imputation in the Model

Missing values in a dataset are a big problem in many areas of statistics. One of the approaches to dealing with missing data is omitting observations containing missing values altogether. The Schistosomiasis dataset has 30.5% of data missing, so this approach would not work.

We used two approaches to deal with the problem of missing data. The first one uses the `MICE` (Multivariate Imputation by Chained Equations) package (van Buuren and Oudshoorn (1999)) for `R` (Ihaka and Gentleman (1996)). Our second approach was to use Gibbs sampling between MH jumps, i.e. sampling missing values using the Gibbs sampler after each MH proposal (Heckerman et al. (1997)).

X_O are observed values and $X_M^{(n)}$ is the set of imputed missing values after n^{th} MH step. We sample a new structure proposal and missing values from the following distributions:

$$\begin{aligned} G_{n+1} \mid X_O, X_M^{(n)} &\sim \frac{p(X_O, X_M^{(n)} \mid G_{n+1})p(G_{n+1})}{p(X_O, X_M^{(n)})} \\ &\propto p(X_O, X_M^{(n)} \mid G_{n+1}) \\ X_{M,i}^{(n+1)} \mid X_O, X_{M,-i}^{(n)}, G_n &\sim \frac{p((X_O, X_{M,i}^{(n+1)}, X_{M,-i}^{(n)} \mid G_n)}{p(X_O, X_{M,-i}^{(n)} \mid G_n)} \\ &\propto p(X_O, X_{M,i}^{(n+1)}, X_{M,-i}^{(n)} \mid G_n) \end{aligned} \tag{2.4.1}$$

This is indeed a full MCMC sampler, as $p(G)$ is uniform for all G, $p(X_O, X_M^{(n)})$ is uniform for all data and $p(X_O, X_{M,-i}^{(n)} \mid G_n)$ is constant for all $X_{M,i}^{(n+1)}$ between MH steps

(here $X_{O,i}^{(n+1)}$ is the i^{th} missing value at $(n+1)^{st}$ step and $X_{O,-i}^{(n)} = \{X_{O,1}^{(n+1)}, ..., X_{O,i-1}^{(n+1)}, X_{O,i+1}^{(n)}, ..., X_{O,N_{missing}}^{(n)}\}$), i.e. $1, ..., i-1$ missing values sampled in $(n+1)^{st}$ step and $i+1, ..., N_{missing}$ missing values sampled in n^{th} step.

Our Markov chain samples in the space

$$\mathcal{G} \times \underbrace{\{1,2\} \times ... \times \{1,2\}}_{N_{\text{missing}} \text{ times}} \tag{2.4.2}$$

where $\mathcal{G}$ is the space of all possible structures, N_{missing} is the total number of the missing values and each node can take values in a set $\{1, 2\}$.
We start from some random point $\{G_1, x_1, ..., x_{N_{missing}}\}$, i.e. some starting structure G_1 and some starting value x_i for each missing value $X_{M,i}$, $i = 1, ..., N_{\text{missing}}$. Then in each step $n = 1, 2, ...$ we propose a structure change $G_* \in \mathcal{G}$. If the change is accepted we change the structure into G_*, otherwise we keep the old structure G_n, then we cyclically resample a new value $x_i \in \{1, 2\}$, for each missing value $X_{M,i}$, $i = 1...N_{\text{missing}}$.

2.5 Parallel Tempered MCMC

2.5.1 Issues with Chain Mixing

Given that the Markov chain transition space is irreducible, the Metropolis-Hastings algorithm is guaranteed to approximate any distribution it is sampling from, provided it runs for long enough. But this is a theoretical result and it only says that we will get a reliable result assuming we can wait as long as necessary. In practise, it could take too long for the chain to converge. In order for the chain to converge quickly, it needs to be able to make sufficiently big steps and have a good average proposal acceptance rate. The traditional approach might not mix very well because it can get stuck in a local mode or have a low average proposal acceptance rate.
A chain is especially likely to get stuck in a local mode in high dimensions, as the modes of a distribution can be separated by large regions of extremely low probability. The probability of accepting a step into this low probability region is very small, therefore, once a chain enters a local mode it can take it a very long time to escape. Increasing the step size of the proposal could increase its chance of escaping from a local mode (by jumping over the low probability region altogether), but that reduces the acceptance probability for an average jump even further.

The second possible issue is a low average proposal acceptance rate. Usually this problem can be tackled by reducing the step size, so that the current state and the proposal come closer together and, therefore, the jump is more likely to be accepted. Our model is prone to suffer from both issues. First of all we are working in a high dimensional space so the probability distribution is very likely to have sharp peaks. Furthermore, due to the discrete nature of our search space, the step size is fixed; the newly proposed structure differs from the current structure by at least one edge. As a result, it is impossible to decrease the step size in order to increase the acceptance probability.

2.5.2 Tempered MCMC

It can be difficult to use Metropolis-Hastings to sample from a distribution with very sharp peaks, because if a chain enters a local maximum it is very likely to get stuck there.

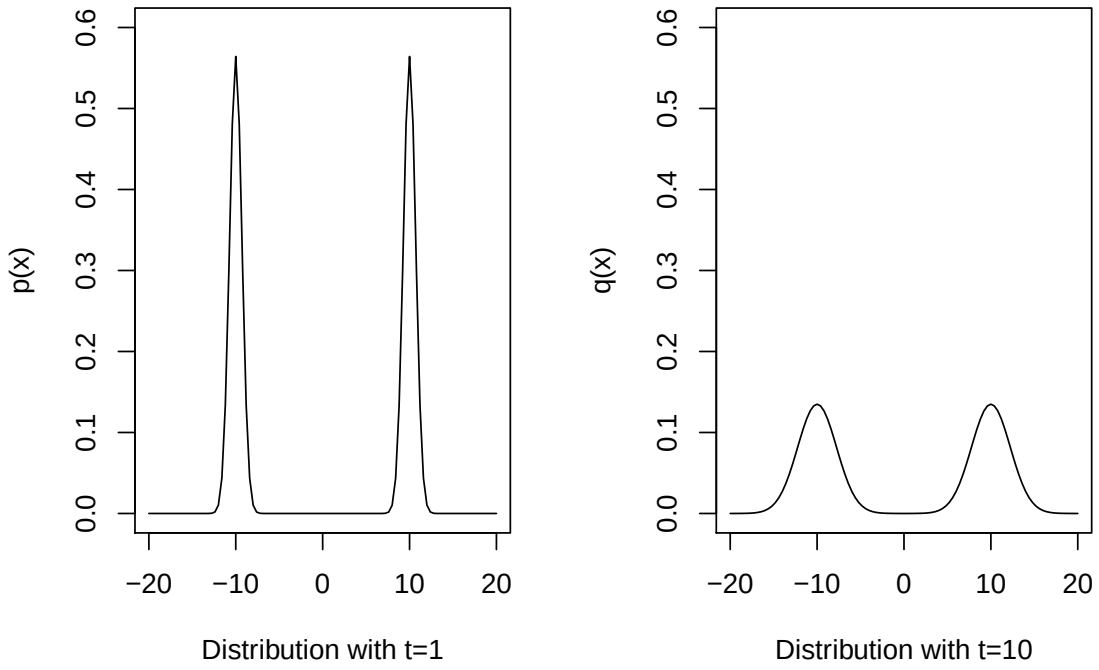


Fig. 2.3 On the left: bimodal distribution $p(x)$; on the right: tempered distribution $q(x) \propto p(x)^{\frac{1}{10}}$, with temperature 10

Consider the example in Figure 2.3. The distribution $p(x)$ has very sharp peaks at $x = -10$ and $x = 10$. MH is likely to get stuck in one of the modes and it can take a very long time to find the other mode. It would be difficult to sample from $p(x)$ using MH. The distribution $q(x)$, on the other hand, is much flatter and it would be much easier for a MH sampler to sample from it. $q(x)$ has the same modes as $p(x)$, therefore sampling from $q(x)$ can help us find all the modes of $p(x)$.
The underlying idea is that we can use a temperature parameter to "flatten" out the distribution. MCMC will move and mix much faster on this flatter probability distribution.
We can write a probability density as $p(x) = \exp(-E(x))$, where $E(x)$ is the energy of the system. For a given temperature t, probability distribution is $p(x,t) = \exp(-E(x))^{1/t} = \exp(-E(x)/t)$. We say that we run the MCMC chain at the temperature t.

2.5.3 Parallel Tempered MCMC

An MCMC chain with a temperature $t = 1$ will be sampling from the correct distribution, but will explore the space slowly and will be prone to getting stuck in local modes. An MCMC with a higher temperature will explore the space faster and have a better average proposal acceptance rate, but it will not in fact be sampling from the distribution we are actually interested in.
The parallel Tempered MCMC algorithm (Iba (2001), Earl and Deem (2008)) allows us to deal with both of these issues. The idea behind the parallel tempered MCMC is to run multiple chains at different temperatures. After a certain number of iterations, we will propose a "sweep", that is a chance for the chains to swap the states they are currently in among themselves. This might propose a better state far away for the lower temperature chain (which was found by the fast moving high temperature chain). We are proposing the states swapping according to a detailed balance equation.

Here $\{t_m, m = 1, ..., M\}$ is a set of temperatures, such that $1 = t_1 < t_2 < ... < t_M$. N_{sweep} is the number of sweeps. N is the number of iterations between sweeps. $x_{i,j}^{(m)}$ is the state of the MCMC chain at the temperature t_m at the j^{th} iteration after the $(i-1)^{st}$ sweep.
We initialize M chains and run them at different temperatures. After every N steps we perform a so-called sweep; that is we consider swapping the current states of the chains. We propose to swap only the adjacent chains (chains with temperatures t_m and t_{m+1} for some $m = 1, ..., M-1$). This is done in order to improve the swap acceptance

Data: X - data, T - number of chains, N_{sweep} - number of sweeps, N - number of iterations per sweep, $\{t_1, ..., t_T\}$ - temperatures
Result: MCMC sample
initialize starting points $x_{0,N}^{(m)}$, for $m = 1...M$;
for $i = 1 : N_{sweep}$ **do**
  for $\tau = 1 : T$ **do**
    assign $x_{i,0}^{(\tau)} = x_{i-1,N}^{(\tau)}$;
    run the chain $p(x, t_\tau)$ at the temperature t_τ using MH for N steps starting with $x_{i,0}^{(\tau)}$ generating $x_{i,N}^{(\tau)}$
  end
  for $\tau = 1 : T - 1$ **do**
    swap $x_{i,N}^{(\tau)}$ with $x_{i,N}^{(\tau+1)}$ with probability $\alpha_\tau = \min\left(1, \frac{p(x_{i,N}^{(\tau+1)}, t_m) p(x_{i,N}^{(\tau)}, t_{m+1})}{p(x_{i,N}^{(\tau)}, t_m) p(x_{i,N}^{(\tau+1)}, t_{\tau+1})}\right)$
  end
end
return $\{x_{i,j}^{(1)} : i = 1, ..., N_{sweep}, j = 1, ..., N\}$

Algorithm 1: Parallel Tempered MCMC

rate (if the temperatures are very different a swap is very unlikely to be accepted). We swap between the chains of temperatures t_m and t_{m+1} with the probability α_m. We should note, that probability α_m is analogous to the probability with which we would accept simultaneous independent jumps $x_{i,N}^{(m)} \to x_{i,N}^{(m+1)}$ in the chain of temperature t_m and $x_{i,N}^{(m+1)} \to x_{i,N}^{(m)}$ in the chain of temperature t_{m+1} proposed in the usual MH setup.

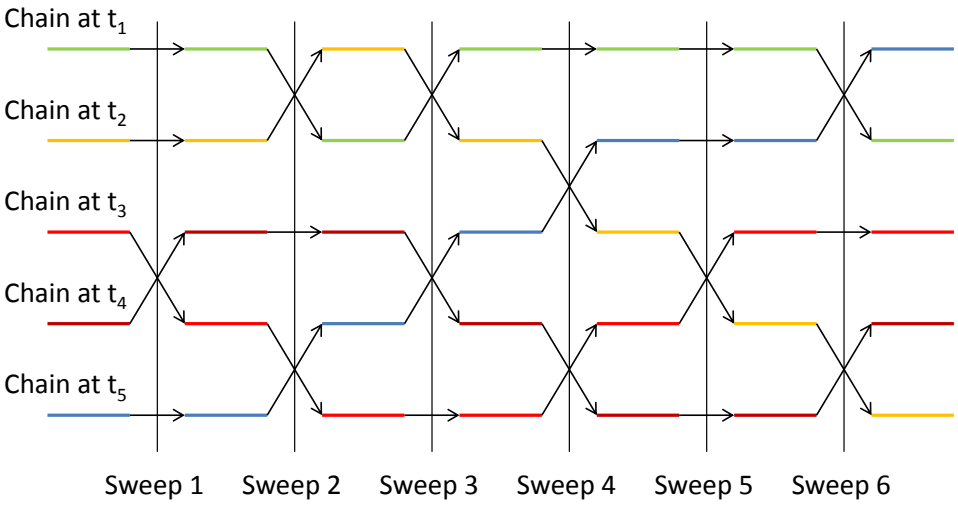


Fig. 2.4 Example of running 5 chains at temperatures $t_1, ..., t_5$ for 6 sweeps

Figure 2.4 shows an example of a parallel tempered MCMC, and we can observe how over time the information from a high temperature chain 5 can be transmitted to a low temperature chain 1.

2.5.4 Temperatures

Choosing the set of temperatures $\{t_m, m = 1, ..., M\}$ is a difficult problem. The chains should mix well amongst themselves; it is important that mixing is not limited to neighbouring chains only, as this would result in some chains never reaching the lowest or highest temperatures. To accomplish this, the difference in the consecutive temperatures t_m and t_{m+1} cannot be too large for all $m = 1, ..., M-1$. Furthermore, the chain with the highest temperature needs to mix quickly, so the difference between t_1 and t_M should be big enough to ensure that. As the complexity of the algorithm grows linearly in M, we don't want to accumulate too many chains. Satisfying all of these conditions can be difficult. We used a geometric progression of temperatures, i.e. $t_m = \tau^{m-1}$, for some τ. This approach is easy to implement and has been proven to give good results (Earl and Deem (2008)). For these temperatures α_m simplifies to:

$$\begin{aligned}
\alpha_m &= \min\left(1, \frac{p(x^i_{m+1}, t_m)p(x^i_m, t_{m+1})}{p(x^i_m, t_m)p(x^i_{m+1}, t_{m+1})}\right) \\
&= \min\left(1, \frac{\exp(-E(x_{m+1})/t_m)\exp(-E(x_m)/t_{m+1})}{\exp(-E(x_m)/t_m)\exp(-E(x_{m+1})/t_{m+1})}\right) \\
&= \min\left(1, \exp((E(x_{m+1} - E(x_m))(\frac{1}{t_m} - \frac{1}{t_{m+1}}))\right) \\
&= \min\left(1, \exp((E(x_{m+1} - E(x_m))(\frac{\tau - 1}{\tau^m}))\right)
\end{aligned}$$

In our case $E(G) = -l(G \mid n, \alpha) = -\log \Pr(n \mid G, \alpha)$.
There are other ways of choosing the temperatures. For example, we could change the temperatures dynamically, in order to keep swap acceptance high and similar at all levels (Wang and de Freitas (2011), Hamze et al. (2010)).

2.6 Results

In this section we will discuss how our algorithm works in practice. First we will cover how it works on small (6 nodes) simulated networks. These networks were chosen to demonstrate the most representative structures that can be observed in the

networks, namely: a chain (or a linear structure), unshielded or not unshielded colliders (a collider with respectively marginally independent or dependent parents) and tree structure. The real datasets that we will investigate are with 11 nodes, therefore we will continue with a few examples of bigger (11 nodes) networks. Finally we will present the results on the real and simulated Sachs datasets (Sachs et al. (2005)) (where we have a gold-standard to evaluate our results) and the Schistosomiasis (where we do not have a reference network to evaluate our results) dataset. Finally we will explore the full (28 variable) Schistosomiasis dataset.

2.6.1 Small Examples

We started by testing our method on a few simulated examples. We generated 1000 data points from a Bayesian network and ran our algorithm in order to recreate the network which generated this data. These are the results we got from running our algorithm on four Bayesian networks on six nodes each.

Chain graph

The first example is a chain graph G_{Ch}; its structure is shown in Figure 2.5a. It should be noted that when we use the likelihood equivalence preserving priors, graphs $G_0, G_1, ..., G_5$ all have equal likelihoods:

$$G_0 = X_1 \to X_2 \to X_3 \to X_4 \to X_5 \to X_6$$

$$G_1 = X_1 \leftarrow X_2 \to X_3 \to X_4 \to X_5 \to X_6$$

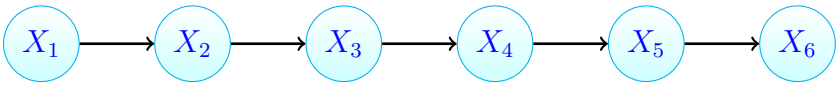
$$...$$

$$G_5 = X_1 \leftarrow X_2 \leftarrow X_3 \leftarrow X_4 \leftarrow X_5 \leftarrow X_6$$

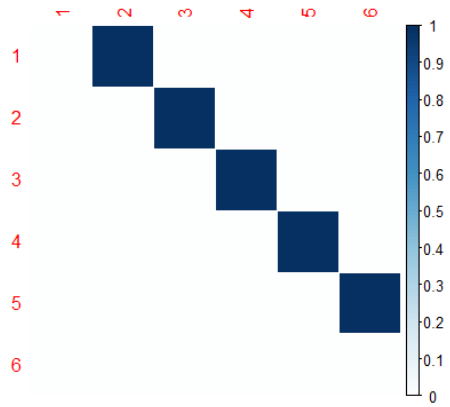
The graph structure probability distribution on the dataset generated from G_{Ch} has a very sharp peak on the subset $\mathcal{G}^* = \{G_i, i = 0, ..., 5\}$. So when our algorithm found this subset it remained there for most of the time. It is worth noting that MCMC will move through $\mathcal{G}^*$ in a similar fashion to a symmetric random walk on $\{0, 1, ..., 5\}$. This is exactly the result we see in Figure 2.5.

Tree graph

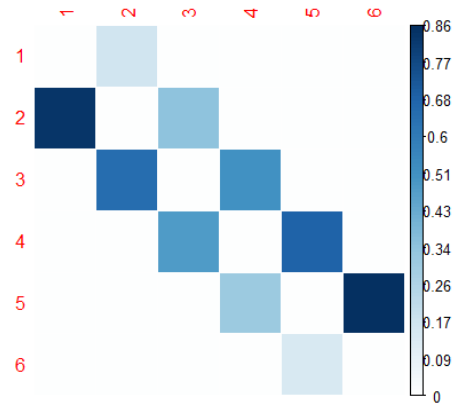
The second example is a tree graph G_T; its structure is shown in Figure 2.6a. We should note that, when likelihood equivalence preserving priors are used, the edges



(a) Chain graph G_{Ch} on 6 nodes



(b) The original Bayesian network



(c) The Bayesian network found by our algorithm

Fig. 2.5 Adjacency matrix for the original graph G_{Ch} and the output of MCMC sampler. Matrix M element m_{ij} represents the probability with which the i^{th} node is a parent of the j^{th} node.

$e_{12} = \{v_1, v_2\}$ and $e_{13} = \{v_1, v_3\}$ can be reversed one at a time, but not together, without creating a collision node (so the conditional independence properties remain the same), i.e. without changing the likelihood of the graph. This is exactly the result we see in fig. 2.6: MCMC found the mode but was unable to determine the direction of e_{12} and e_{13}.

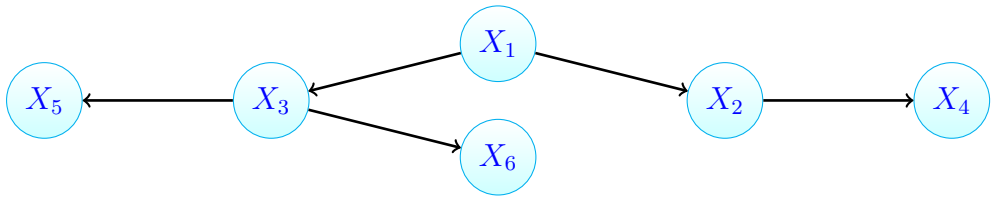
Collision graph

The third example is a 'collision' graph G_C; its structure is shown in Figure 2.7a. In this graph, all three nodes which have parents are collision nodes. This suggests that our algorithm should have no issues in finding the exact graph which generated the data.

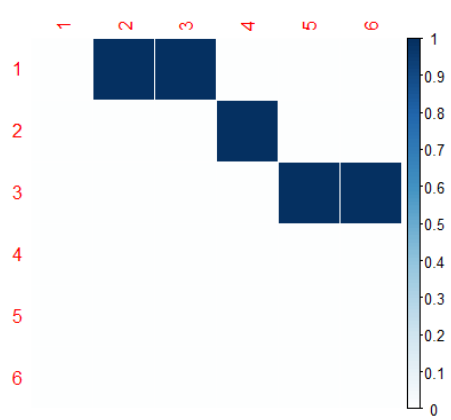
Note: if Z is a collision node, i.e. we have a structure $X \to Z \leftarrow Y$, and we have data n generated from this structure, then

$$l(X \to Z \leftarrow Y \mid n) \gg l(X \leftarrow Z \leftarrow Y \mid n) = l(X \to Z \to Y \mid n) = l(X \leftarrow Z \to Y \mid n)$$

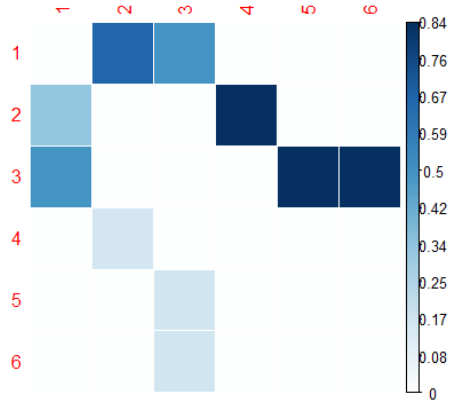
We can see from Figure 2.7, that our algorithm did not, in fact, have any problem finding G_C and that there is no ambiguity about the direction of the edges in this case.



(a) Tree graph G_T on 6 nodes



(b) The original Bayesian network



(c) The Bayesian network found by our algorithm

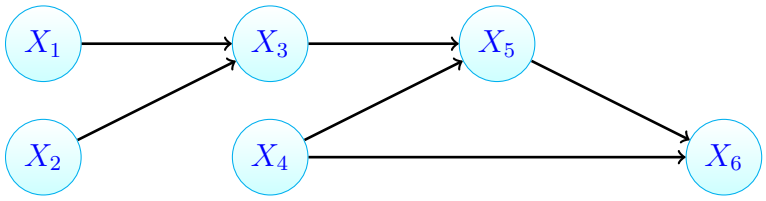
Fig. 2.6 Adjacency matrix for the original graph G_T and the output of MCMC sampler.

Triangle graph

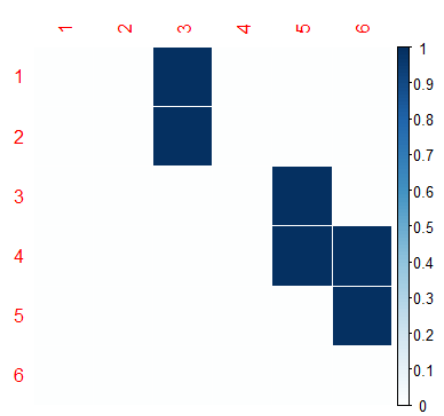
The last example is a 'triangle' graph G_{Tr}; its structure is shown in fig 2.8a. In this graph the node v_6 is a collision node with marginally independent parents, while node v_3 is a collision node with non-independent parents, i.e. the nodes v_1, v_2, v_3 create a triangular structure. Any non-cyclic orientation of edges between these three nodes yields the same likelihood, therefore we expect our algorithm to have difficulty determining the orientation of edges between v_1, v_2, v_3. The result produced by our algorithm is exactly what we expected (shown in fig. 2.8). It found the connected subsets, determined the orientation of edges $\{v_4, v_6\}$ and $\{v_5, v_6\}$, but not the orientation of edges between the nodes v_1, v_2, v_3.

2.6.2 11-Variable Examples

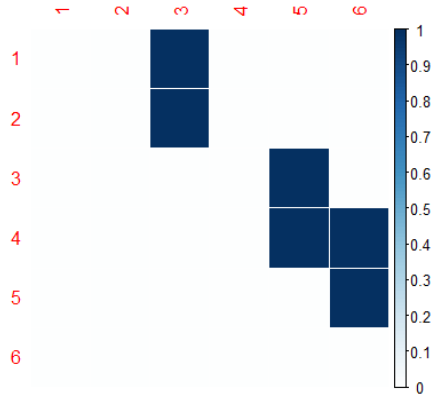
Our original dataset (reduced case) has 11 variables, so it is important to check how well our algorithm can deal with networks of 11 nodes. We tested it on a few networks of 11 nodes. First we generated a random network structure and sampled 1000 data



(a) Collision graph G_C on 6 nodes



(b) The original Bayesian network



(c) The Bayesian network found by our algorithm

Fig. 2.7 Adjacency matrix for the original graph G_C and the output of MCMC sampler.

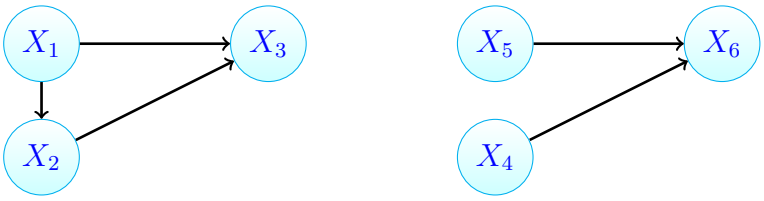
points from it. We then ran our algorithm to recreate the network: Figures 2.9, 2.10 and 2.11 show three examples of the results.

The relatively simple structures in Figure 2.9 and Figure 2.10 meant that our algorithm was able to reconstruct the networks up to the direction of a few edges. The edges $\{v_{11}, v_5\}$ and $\{v_8, v_7\}$ in Figure 2.9 and $\{v_9, v_7\}$ in Figure 2.10 are independent of all the other edges. Therefore, the likelihood of the network is not affected by their directionality, i.e. our algorithm will not be able to identify their direction if we use likelihood equivalence preserving priors.

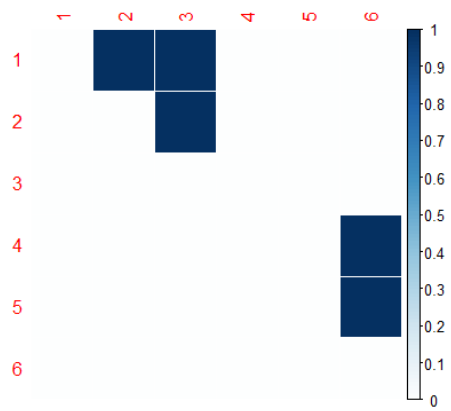
The network structure in Figure 2.11 is more complicated. As a result, our algorithm is only able to determine clusters. This is expected, as the subsets of nodes $\{v_5, v_{10}, v_{11}\}$ and $\{v_3, v_4, v_6\}$ form 'triangle' structures (discussed in Section 2.6.1).

2.6.3 Conclusions

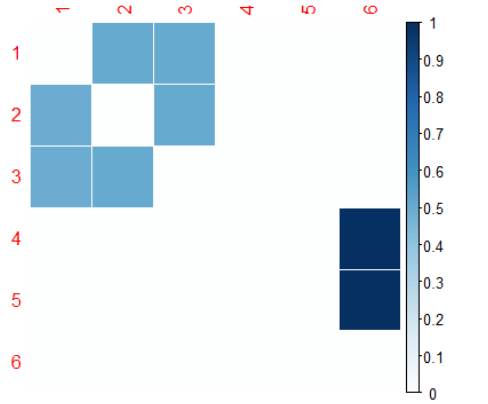
These four examples were chosen because they cover the main possible edge configurations encountered in Bayesian networks. We observed that our algorithm is very efficient in recognising collision nodes when the parents are marginally independent



(a) Triangle graph G_{Tr} on 6 nodes

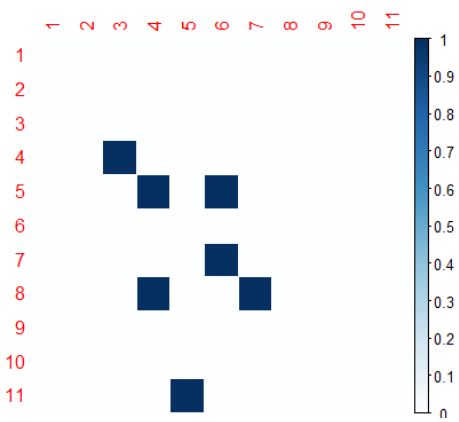


(b) The original Bayesian network

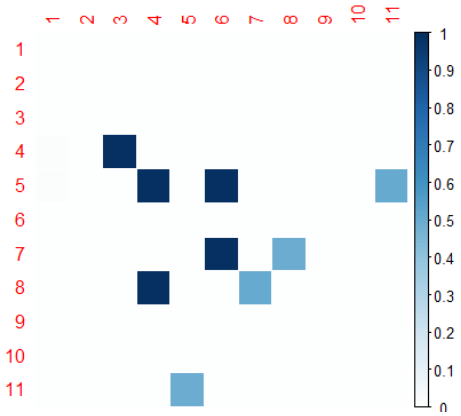


(c) The Bayesian network found by our algorithm

Fig. 2.8 Adjacency matrix for the original graph G_{Tr} and the output of MCMC sampler.



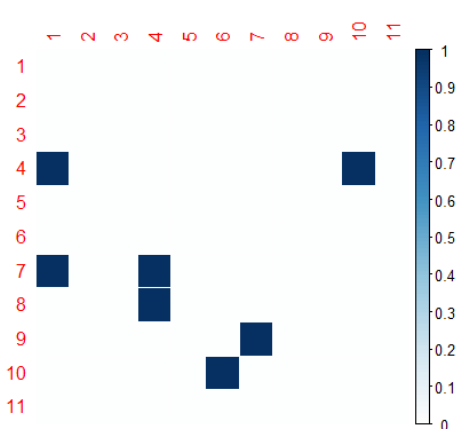
(a) The original Bayesian network



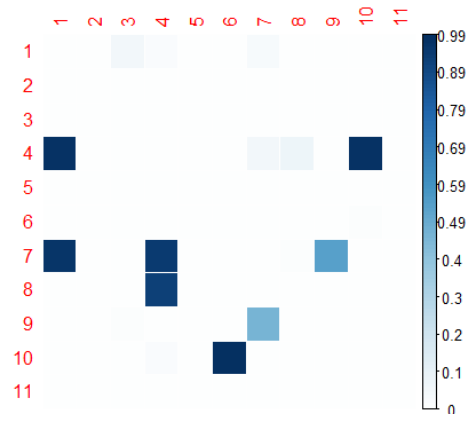
(b) The Bayesian network found by our algorithm

Fig. 2.9 Adjacency matrix for the original graph on 11 nodes G_1 and the output of MCMC sampler.

and can effectively find chains and triangular structures. However, as expected, it had no efficient way of determining the orientation of edges in these structures.

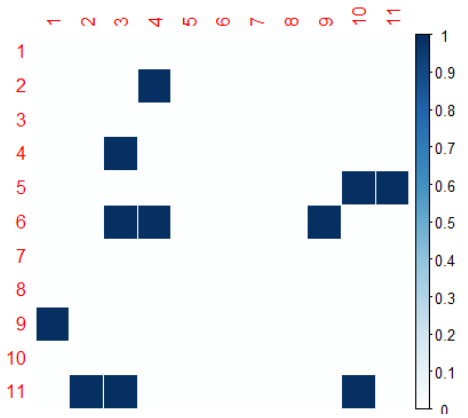


(a) The original Bayesian network

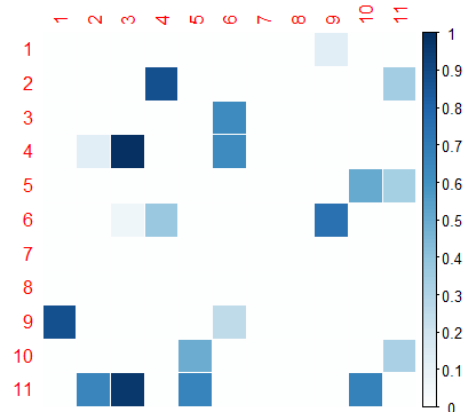


(b) The Bayesian network found by our algorithm

Fig. 2.10 Adjacency matrix for the original graph on 11 nodes G_2 and the output of MCMC sampler.



(a) The original Bayesian network



(b) The Bayesian network found by our algorithm

Fig. 2.11 Adjacency matrix for the original graph on 11 nodes G_3 and the output of MCMC sampler.

2.6.4 Single-Cell Datasets

For real datasets we start with the single-cell datasets from Sachs et al. (2005). These are convenient because we have a reference network of "well known" dependencies from the literature which we can use to evaluate our results. For brevity we present only the results from Dataset 8 here. Results for the remaining datasets can be found in the Appendix, Section B Figure B.1.

First we used the original single-cell Dataset 8 to generate 100 simulated datasets using the reference network as described in Section 1.6.2. Then we ran an MCMC

simulation on each of these datasets. To evaluate the results we used the receiver operating characteristic (ROC) curve. The MCMC run gives us the probability of each edge being in the network. Then we order the edges from the most likely to be included in the network to the least likely and compare this ordering to the reference network (i.e. we start with an empty network and add one edge at a time according to this ordering and compare that network with the reference network). Note that in this case we are comparing the undirected edges only, that is, the edges of the skeleton. Each such network gives us a certain number of true/false positives/negatives which allows us to create a ROC curve. In Figure 2.12 we present the 100 ROC curves generated in this way. The 100 runs give us an area under a curve of 0.82 ± 0.04.

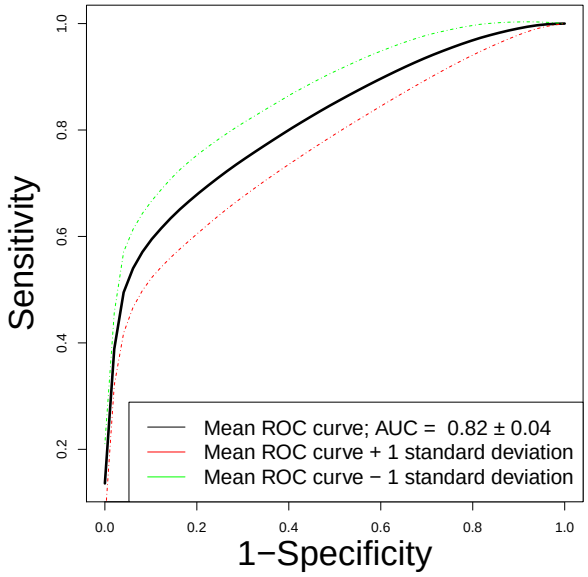
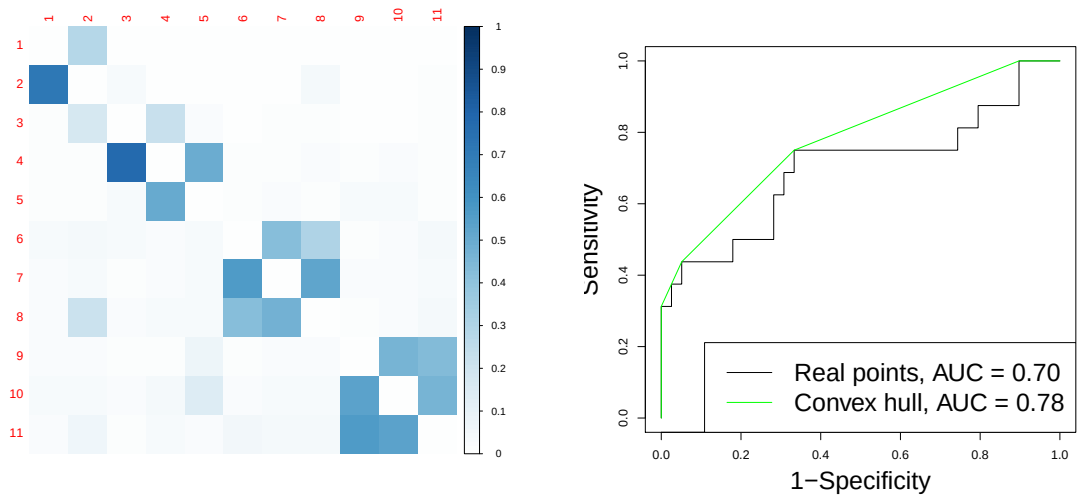


Fig. 2.12 Results from datasets simulated from the Sachs Dataset 8.

Next we ran the algorithm on the original single-cell Dataset 8. Results are provided in Figure 2.13. As expected the area under the ROC curve is slightly smaller than in the simulated dataset case. This is to be expected as the original dataset is not necessarily generated from the DAG. The result is still quite good and in the range of one standard deviation from the mean of the area under the ROC curves from the simulated datasets. Therefore the algorithm does not perform significantly worse on the real data than on the simulated one.

2.6.5 Schistosomiasis Dataset

In this section we will demonstrate how our algorithm is able to deal with the Schistosomiasis dataset. Recall that Schistosomiasis data comes from known time slices, i.e.



(a) Probabilities of each edge being in the final model.

(b) ROC curve.

Fig. 2.13 Results from the Sachs Dataset 8.

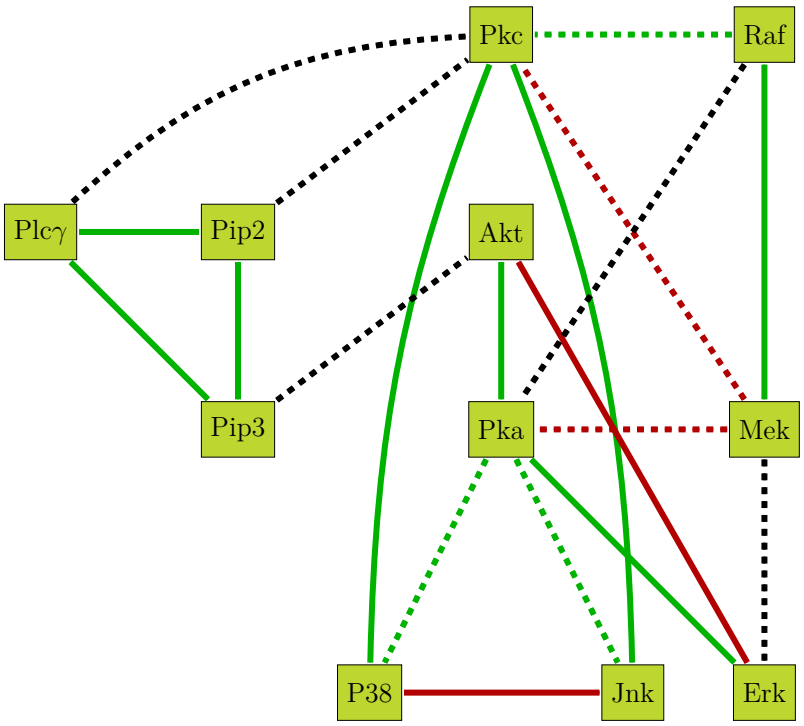
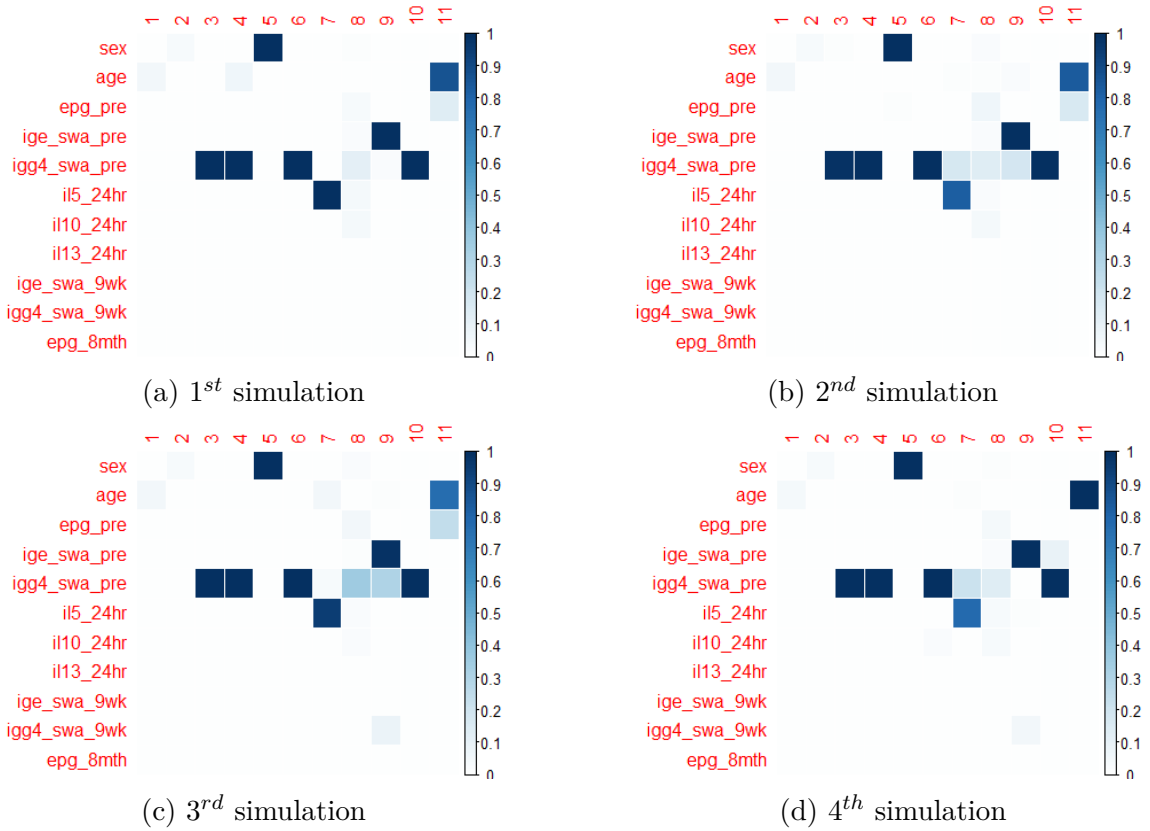


Fig. 2.14 Network from the single-cell data generated by the MCMC using the discretized data. Green are the correctly identified edges, red are wrongly identified edges and dashed black are the edges that the algorithm did not find. Solid lines represent edges with high probability and dashed lines represent edges with low probability.

before the treatment, 24h after the treatment, etc. We use this as a prior knowledge by only allowing edges going from an earlier time slice to a later time slice but not the other way around. In this dataset, 17.4% of all the values are missing. We used two different approaches to deal with this (covered in detail in Section 2.4). Our first approach used an MH algorithm with Gibbs sampling of the missing values between MH steps. The results are shown in Figure 2.15.



(a) 1^{st} simulation

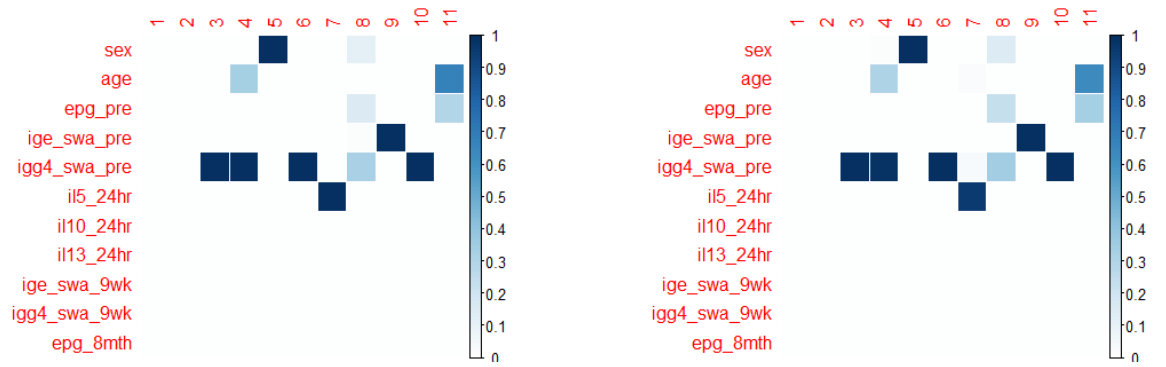
(b) 2^{nd} simulation

(c) 3^{rd} simulation

(d) 4^{th} simulation

Fig. 2.15 The Schistosomiasis network adjacency matrix found by our algorithm using Gibbs sampling for the missing value imputation. Matrix M element m_{ij} represents the probability with which the i^{th} node is a parent of the j^{th} node.

For the second approach we use the `MICE` package for `R` to create 50 datasets with imputed values, run a simple MH algorithm on all of them, and finally pool the results. We chose 50 datasets because this number of simulations can be done in a reasonable amount of time and still provides a consistent result. In Figure 2.16 we provide the result given by aggregating distributions from the first and second sets of twenty-five imputed datasets. As we can see, the two distributions are almost identical (the order of the 13 most likely edges is the same).



(a) Aggregate result from the first 25 simulations

(b) Aggregate result from the second 25 simulations

Fig. 2.16 The Schistosomiasis network adjacency matrix found by our algorithm using the MICE package for the missing value imputation. Matrix M element m_{ij} represents the probability with which the i^{th} node is a parent of the j^{th} node.

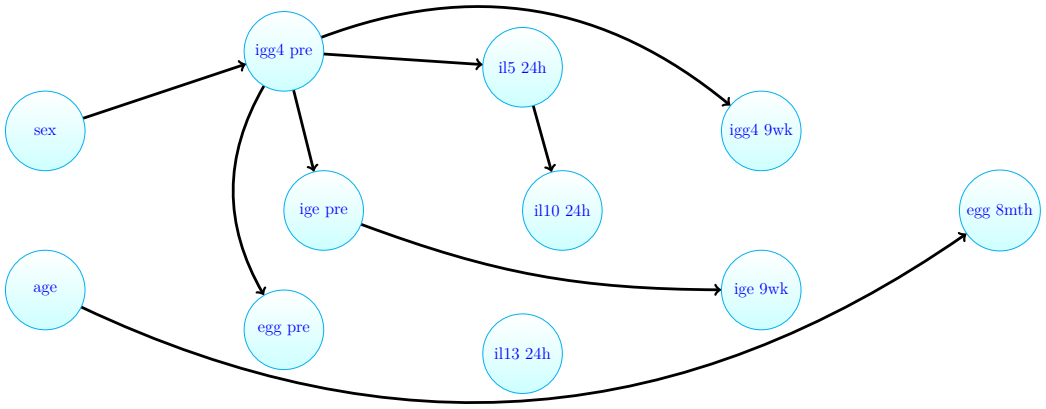


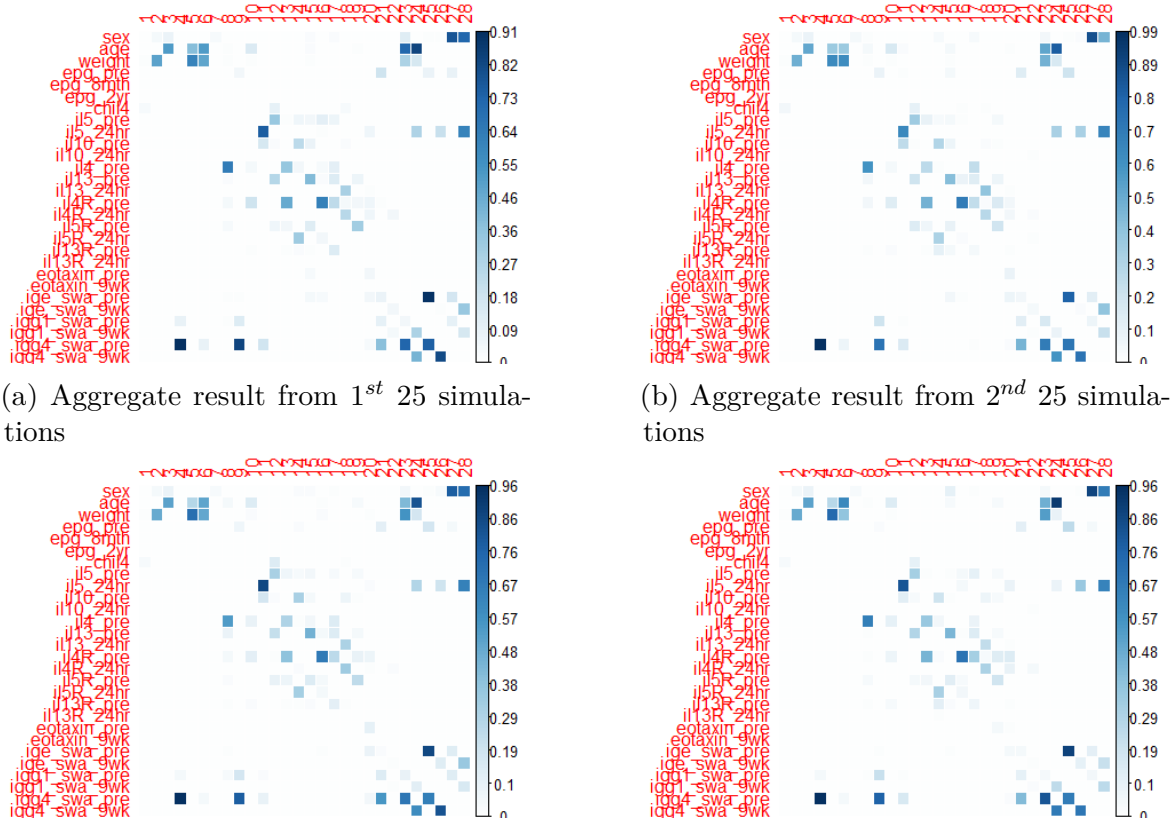
Fig. 2.17 Network for the Schistosomiasis dataset; only the edges with probability above 0.5 are shown.

The first approach samples from the correct distribution. But, because it needs to resample all of the missing values each time the MH proposal is accepted, it takes much longer than a simple MH algorithm on a dataset with no missing values, such as the second approach. Using datasets where missing values are imputed with `MICE` might seem dubious, because it is not necessarily clear if we are sampling from the correct distribution. However we can see from Figures 2.15 and 2.16 that the two approaches give almost the same result (the order of the 8 most likely edges is the same). This suggests that the second approach, if not absolutely rigorous, is at least a

good approximation. A graph consisting of edges with probabilities above 0.5 is shown in Figure 2.17.

2.6.6 28-Variable Case

Finally, we ran our algorithm on all 28 variables of the Schistosomiasis dataset. It would be inefficient to impute the missing values using the Gibbs sampler here. But, having seen that imputing missing values using `MICE` and then running a simple MH structure search produced good results in the 11-variable case, we used the same approach here. We imputed 100 datasets, ran our algorithm on each of them, and produced four distributions pooled from 25 imputed datasets each. The results are shown in Figure 2.18.



(a) Aggregate result from 1^{st} 25 simulations

(b) Aggregate result from 2^{nd} 25 simulations

(c) Aggregate result from 3^{rd} 25 simulations

(d) Aggregate result from 4^{th} 25 simulations

Fig. 2.18 The Schistosomiasis network adjacency matrix found by our algorithm using the MICE package for the missing value imputation. Matrix M element m_{ij} represents the probability with which the i^{th} node is a parent of the j^{th} node.

We observe that all 4 pooled distributions are very similar, which gives us some degree of confidence that the MCMC did converge. However; very few edges have a high probability, and quite a few edges have rather low, but still not negligible probabilities. This suggests that in order to deal with networks this size, either our approach needs to be altered, or additional post-processing is required.

2.6.7 Scaling to Larger Networks

In this thesis we are considering two experimental datasets both having 11 nodes. In genomics these are quite modest sized datasets. In order to give an idea of how this algorithm scales to larger networks we will consider networks on 20, 50 and 100 nodes. All simulations are run using code implemented in `R` and using Intel(R) Core(TM) $i7-7700HQ$ CPU @ 2.80GHz.
Example 1: Random network G_{20} on 20 nodes, 300 or 800 simulated observations from G. Ran the MCMC search algorithm for 10^4, 10^5 and 10^6 iterations. Averages out at approximately 7 seconds per 10^4 iterations, almost no difference in time taken to run the algorithm when using dataset with 300 or 800 data points.
Example 2: Random network G_{40} on 50 nodes, 300 simulated observations from G. Ran the MCMC search algorithm for 10^4, 10^5 and 10^6 iterations. Averages out at approximately 9 seconds per 10^4 iterations.
Example 3: Random network G_{100} on 100 nodes, 300 simulated observations from G. Ran the MCMC search algorithm for 10^4, 10^5 and 10^6 iterations. Averages out at approximately 14 seconds per 10^4 iterations.
For full details see Table 2.3. From these examples we can observe that the number of the observations does not impact the speed of the algorithm too much. On the other hand the size of the network has a strong effect on the computational time. It is important to note that it takes significantly more iterations of the MCMC to achieve convergence and get a good AUC for the larger networks. This imposes a natural time constraint on the viability of the algorithm, if we have a particularly large network it may take infeasibly long to achieve the convergence of the MCMC. This is not specific to this algorithm - convergence of the MCMC is a well known problem.
Few other considerations:
1) It is important to note that saving a full MCMC simulations is not very efficient in terms of memory: even in the relatively small example of a graph on 20 nodes and taking 10K iterations we would need to store a matrix $20 \times 20 \times 10^4 \times 48$B (20×20 for each adjacency matrix, 10^4 each iterations, 48 Bytes to store a float number in

Example	# Nodes	# Samples	# Iterations	Time taken	AUC
1a	20	300	10^5	65s	0.86
1b	20	300	10^6	12min	0.92
1c	20	800	10^5	70s	0.88
1d	20	800	10^6	12.5min	0.97
2a	50	300	10^4	8.8s	0.76
2b	50	300	10^5	1.5min	0.83
2c	50	300	10^6	14.2min	0.93
3a	100	300	10^4	18.1s	0.65
3b	100	300	10^5	2.4min	0.82
3c	100	300	10^6	23.4min	0.89

Table 2.3 Time taken and the mean Area Under the Curve achieved by the BN MCMC algorithm for networks with various number of nodes and various number of observations.

`R`) which would require almost 2GB of memory. In order to avoid this we are using thinning of the MCMC paths saving only every k^{th} value.

2) Another issue is a dense graph, current implementation is storing a graph object which contains information about the conditional probabilities of the child node, given its parents nodes. We need to store $2^{\|\mathrm{Pa}_G(X_i)\|}$ conditional probabilities, where $\mathrm{Pa}_G(X_i)$ is the set of parents for the node X_i. A constraint on the number of parents is required to prevent the memory explosion.

3) Using n cores this algorithm can be parallelized to achieve an almost n-fold increase in the computation time. The less than n-fold increase is especially prominent in case we are using the parallel tempered MCMC. In this case the chains need to communicate at every sweep. This requires all the cores to finish there calculations, gather the results to reorder the chains and then send the jobs back to separate cores. On the other hand if we are running n chains at the same temperature we can just run all of them and gather the results at the end, this would provide an almost n-fold increase in computational speed as minimal extra actions are required: only one-off sending jobs to separate cores and only one-off collecting all the completed jobs at the end.

4) The computational estimates are based on the current implementation of the algorithm in `R`. `R` is not a compiled language and therefore it is significantly slower than say `C++`.

2.7 Discussion

We tested our algorithm on both simulated datasets and the experimental single-cell (Sachs et al., 2005) and Schistosomiasis datasets (the reduced, 11-variable case and the full, 28-variable case).
Our algorithm dealt with modestly-sized networks of up to 11 nodes reasonably well. It was also able to deal with a large percentage of missing data. Larger networks caused our algorithm some trouble, and further work is still required in order to find a satisfactory network for the entire Shistosomiasis dataset with its 28 variables.
It is important to note that out algorithm had no proper means of determining the directionality of edges in a chain or a triangular structure. This means that we are only able to identify connected cliques, but still have no accurate method for inferring the causality within them. This did not present much of an issue for the reduced 11-variable Schistosomiasis dataset, because the partial ordering is quite close to the total ordering (not more than three nodes per time slice). Therefore, the result showed clear edge directionality, despite taking the form of a tree graph. On the other hand, in the single-cell dataset we were only able to identify 4 cliques, but not the edge directionality in them. The same can be said about the full, 28-variable Shistosomiasis dataset, which was significantly more problematic, because there were considerably more nodes in each time slice and this resulted in greater ambiguity in edge directionality.

2.7.1 Model Limitations and Future Extensions

The limitations of this approach are two fold.
First of all this method can only deal with discrete data. Actually it only works with categorical data. If we discretize data into more than two buckets the closed form likelihood would not take the ordering of the buckets into account. Future work would involve extending the method for ordered discrete data and preferably for continuous data. We are dealing with continuous data in Chapter 5 but we don't have a closed form solution for the likelihood in this case.
The second limitation is the size of the network that can be inferred using this algorithm. As we have shown above there is strong relationship between the speed of the algorithm and the size of the network that we can effectively find, i.e. the faster our algorithm runs, bigger networks it can deal with. The first step in increasing the speed of the algorithm would be to reimplement the current code using faster language such as `C++`. Another avenue would be to incorporate a bolder MCMC jumps, for example inverting entire chain subgraph, resampling all the parents of the node, etc. It would

require some additional work in establishing the jump probabilities for these more complex moves (the likelihood of the new graph would be calculated in the same way as now, so to calculate the acceptance probability we would additionally need to find the probabilities for the proposal of the jump and its inverse).

Chapter 3

Independence Criteria

3.1 Introduction

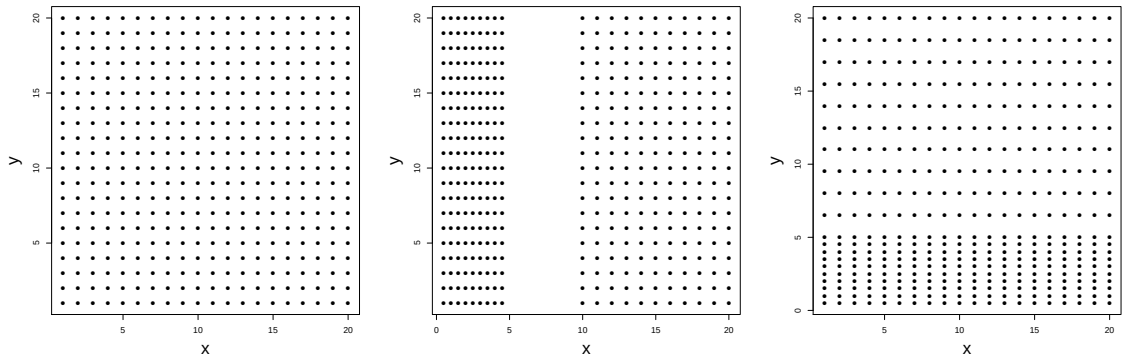
The aim of this chapter is to introduce a quantitative measure of independence of a sample. Later on this measure will be used in Chapter 5. To do so we begin by discussing the idea of independence. Then we present some of the most popular independence criteria, such as Distance Correlation, Kernel Covariance and the Hilbert Schmidt Independence Criterion. We also introduce a new way of estimating the dependence between variables in a special case - the Signal to Noise Ratio Criterion. Finally we discuss how these criteria can be used to produce statistical tests to measure the dependence between the variables. We conclude with a comparison of these criteria on simulated data.

3.1.1 Independence

In order to be able to discus any independence criteria, we need to define independence. Intuitively two events are independent if they do not influence each other.
Suppose we have only two events A and B together with a probability measure p. Then A and B are *independent* (written $A \perp B$) if their joint probability is equal to the product of their probabilities, i.e. $p(A \cap B) = p(A)p(B)$. Now suppose we have a finite set of events $A = \{A_i : i = 1, ..., n\}$. In this case there exists two degrees of independence. We say that A is *pairwise independent* if every pair of events is independent, i.e. $p(A_i \cap A_j) = p(A_i)p(A_j)$ for all distinct $i, j = 1, ..., n$, and that A is *mutually independent* if every event is independent of any intersection of other events, i.e. $p(\bigcap_{i=1}^{k} A_{j_i}) = \prod_{i=1}^{k} p(A_{j_i})$ for all subsets of $\{A_i : i = 1, ..., n\}$. A pairwise independent set of events does not have to be mutually independent.

Example: Consider tossing a fair coin twice. The space of possible outcomes is $\{TT, TH, HT, HH\}$. Let us define three events: $A = \{HT, HH\}$, $B = \{TH, HH\}$, $C = \{TT, HH\}$. Now as the coin is fair $p(A) = p(B) = p(C) = 1/2$. Also $p(A \cap B) = p(A \cap C) = p(B \cap C) = p(\{HH\}) = 1/4$ so all the events are pairwise independent. But $p(A \cap B \cap C) = p(\{HH\}) = 1/4 \neq 1/8 = p(A)p(B)p(C)$ so these events are not mutually independent.

Two random variables X and Y with the probability density functions f_X and f_Y respectively and the joint probability density function $f_{X,Y}$ are independent if and only if $f_{X,Y} = f_X f_Y$. For a finite family of random variables $X = (X_1, ..., X_n)$ the definitions of pairwise and mutual independence extend naturally as described above. In real life applications we usually do not have access to probability density functions, but only to a finite sample from a distribution. Therefore it is important to consider what it means for a finite sample to be truly independent. Suppose we have a finite sample $z = ((x_1, y_1), ..., (x_n, y_n))$ from two random variables X and Y. If the random variables X and Y are independent, having information about one of them does not give us any information about the distribution of the other, i.e. $p(Y \mid X = x_i) = p(Y)$, for all x_i. This will happen only if the finite sample x and y lies on the grid, as in Figure 3.1. If data does in fact lie on a grid as in Figure 3.1a, then for example knowing that, say, $x_1 = 10$ gives us know additional knowledge about y_1, y_1 is equally likely to be any integer from 1 to 20. Note that any squeezing or expanding transformation along the axis (as in Figures 3.1b and 3.1c) of the grid preserves the independence of the data.



(a) Data on an exact grid. (b) Data on a grid: squeezed. (c) Data on a grid: expanded.

Fig. 3.1 Example of truly independent samples.

Suppose we are considering two genes and we are interested in determining whether one of them directly influences another. All we have is a finite sample of measurements

of these protein expression levels. We may treat these two proteins as true underlying random variables and the sample of measurements as the realisations from their distributions. Now the question is "how can we make an inference about whether the underlying random variables are dependent or not, given only the finite sample?" We cannot be absolutely sure about the underlying probability distributions given only the finite sample, but we may create a statistical test which will give us the probability of the underlying random variables being independent given this sample. Trying to construct such a test (and ensuring that it would be correct and efficient) will be the goal of this chapter.

3.2 Distance Correlation

3.2.1 Introduction

In this section we discuss the first independence criteria - the correlation of distance or the Distance Correlation Criterion (dCov) which was introduced in (Székely et al., 2007), a different theoretical justification was later on provided in (Székely et al., 2009). We start from this criteria as it is conceptually the easiest. We start by giving some motivation why this might be a good criteria to look at in order to evaluate the dependence between variables. We will continue by providing a rigorous mathematical justification for this criterion and finally we will provide an empirical estimate that can be used in practice.

3.2.2 Motivation

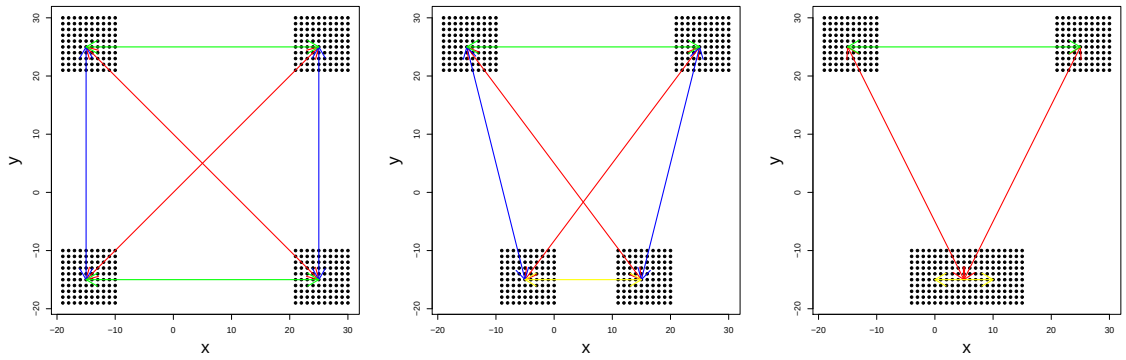
The easiest method to estimate whether a sample $z = ((x_i, y_i), i = 1, ..., n)$ is dependent is to look at the covariance of x and y, i.e. $\mathrm{Cov}(x, y) = n^{-1} \sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})$, where $\bar{x} = n^{-1} \sum_{i=1}^{n} x_i$ is the mean of x, equivalently $\bar{y}$ is the mean of y. We can go one step further and look at the correlation (defined as $\mathrm{Cor}(x, y) = \mathrm{Cov}(x, y)/\sqrt{\mathrm{Cov}(x, x)\mathrm{Cov}(y, y)}$) in order to have a scale-free parameter. After we have calculated the correlation of our sample, we may compare that with expected correlation in case X and Y would be independent and find the probability to observe data as extreme as ours given the underlying independent variables. This provides as with a statistical test to determine whether the underlying random variables are independent or not.
The problem with this approach is that the covariance and similarly the correlation can catch only linear dependencies. Correlation is significantly different from zero if and only if large values of x (significantly greater than $\bar{x}$) go together with large values

of y and small values of x (significantly smaller than $\bar{x}$) go together with small values of y (positively correlated variables); or vice versa, large values of x with small values of y and small values of x with large values of y (negative correlated variables). But what if we have, say, a quadratic relationship: $y = x^2$? Clearly this is not independent, but for x symmetrically distributed around zero, we will find that the correlation of x and y is very close to zero.

Székely et al. (2007) proposed to deal with this problem by looking at the correlation of the distances between the datapoints instead of the correlation of the datapoints themselves. Suppose we have a sample $(x, y) = ((x_1, y_1), ..., (x_n, y_n))$. Instead of looking at the $\text{Cor}(x, y)$ we look at $\text{Cor}(d_x, d_y)$, where $d_x = (\|x_i - x_j\|, i, j = 1, ..., n)$ is an n^2 dimensional vector of Euclidean distances between each pair of x's and d_y is defined in a similar fashion. The exact independence criterion proposed in Székely et al. (2007) is slightly more complicated than just $\text{Cor}(d_x, d_y)$ (we will cover that in the following section), but for a motivation this approximation is good enough.

We may think of the correlation as asking a question: "if x is large (small), is y also large (small)?" While in the set up of the correlation of distances, given two points (x_i, y_i) and (x_j, y_j) the question becomes: "if x_i and x_j are far apart (close together), is y_i and y_j also far apart (close together)?"

In order to make this concept clearer lets consider the data samples in Figure 3.2 and the distances between the datapoints in Figure 3.3. We start with 4 quadrants, each a perfect grid. Data is truly independent in Figure 3.2a and therefore the data in Figure 3.2a also lie on a grid. Then we start moving the two bottom quadrants closer together as shown in Figures 3.2b & 3.2c. Data is no longer independent as knowing y, i.e. whether we are in the top or bottom half, gives us a different distribution of x. Looking just at the correlation of x and y does not tell us anything - the dataset is symmetric with respect to the vertical axis, therefore the correlation is zero. On the other hand if we look at Figure 3.3 we clearly see that the symmetry is being broken and the correlation of distances d_x and d_y tells us that data is not independent any more as the correlation of distances is significantly (we will discuss what "significantly" precisely means and how to quantify that in the following sections) different from zero. So if instead of considering the correlation of original data x and y we consider the correlation of the distances between the datapoints d_x and d_y we get a numerical estimate of how dependent is the data, as the correlation of d_x and d_y will be zero if and only if the data would lie on the grid. This provides us with the first independence criteria: the Distance Correlation Criterion.

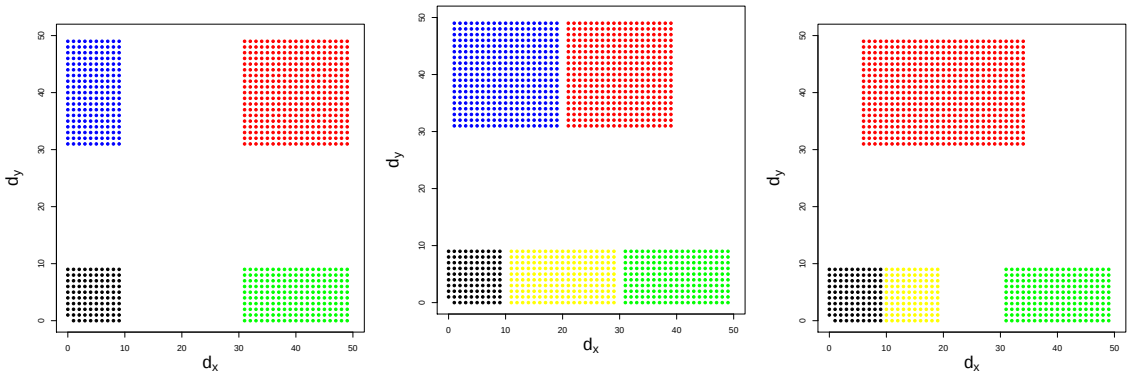


(a) Bottom quadrants apart, $\text{Cor}(x, y) = 0$.

(b) Bottom quadrants closer, $\text{Cor}(x, y) = 0$.

(c) Bottom quadrants together, $\text{Cor}(x, y) = 0$.

Fig. 3.2 Original data divided into 4 quadrants.



(a) Distances of the data from 3.2a, $\text{Cor}(d_x, d_y) = 0$.

(b) Distances of the data from 3.2b, $\text{Cor}(d_x, d_y) = 0.12$.

(c) Distances of the data from 3.2c, $\text{Cor}(d_x, d_y) = 0.24$.

Fig. 3.3 Distances of the data from Figure 3.2. The colour of Section represents the distances along the same coloured arrow in Figure 3.2, for example the green section represents the distances between two quadrants both at the top or both at the bottom, while the black section represents the distances inside the quadrant.

3.2.3 Definition of dCor

So far we have only given a motivation why using the correlation of distances might be a good way of estimating the dependency between two variables. In this section we will provide a short theoretical justification of the dCor, a more detailed explanation is provided in Székely et al. (2007) and Székely et al. (2009).

Suppose we have two random variables $X \in \mathcal{X} \subset \mathbb{R}^p$ and $Y \in \mathcal{Y} \subset \mathbb{R}^q$ (for the majority of the work in this thesis we may assume $p = q = 1$) with characteristic functions $\varphi_X(t) = \text{E}\left[e^{itX}\right]$ and $\varphi_Y(t) = \text{E}\left[e^{itY}\right]$ respectively. Denote the joint characteristic

function of X and Y by $\varphi_{X,Y}(t,s) = \mathrm{E}\left[e^{i(tX+sY)}\right]$. If the random variables X and Y are independent, then $\varphi_X\varphi_Y = \varphi_{X,Y}$ must hold. Therefore in order to check whether X and Y are independent we should test the hypothesis

$$\mathcal{H}_0 : \varphi_{X,Y} = \varphi_X\varphi_Y \qquad \text{vs.} \qquad \mathcal{H}_1 : \varphi_{X,Y} \neq \varphi_X\varphi_Y \tag{3.2.1}$$

To formalise this idea we introduce some terminology. For any function $f : \mathbb{R}^p \times \mathbb{R}^q \to \mathbb{R}$, the *weighted norm* (written as $\|\cdot\|_w$) in the weighted L_2 space of the functions on $\mathbb{R}^{p+q}$ is defined by

$$\|f(t,s)\|_w^2 = \int_{\mathbb{R}^{n+m}} |f(t,s)|^2 w(t,s)dtds,$$

where $w(t,s)$ is an arbitrary strictly positive everywhere weight function for which the integral above exists.
Székely et al. (2007) defines a measure of dependence $\mathcal{V}$ between the random variables X and Y as

$$\begin{aligned}\mathcal{V}^2(X,Y;w) &= \|\varphi_{X,Y}(t,s) - \varphi_X(t)\varphi_Y(s)\|_w^2 \\ &= \int_{\mathbb{R}^{p+q}} |\varphi_{X,Y}(t,s) - \varphi_X(t)\varphi_Y(s)|^2 w(t,s)dtds,\end{aligned} \tag{3.2.2}$$

such that $\mathcal{V}^2(X,Y,w) = 0$ if and only if X and Y are independent. This $\mathcal{V}$ is analogous to the absolute value of the covariance measure. Equivalently to the correlation measure we can define $\mathcal{R}_w$:

$$\mathcal{R}_w = \frac{\mathcal{V}(X,Y,w)}{\sqrt{\mathcal{V}^2(X,X,w)\mathcal{V}^2(Y,Y,w)}} \tag{3.2.3}$$

It is important to note that not all the weight functions w give "interesting" results. We want $\mathcal{R}_w$ to be scale invariant, that is, invariant with respect to transformations $(X,Y) \to (\epsilon X, \epsilon Y)$ for any $\epsilon > 0$. Scale invariance eliminates the necesity to consider data normalization. $\mathcal{R}_w$ must also be positive for dependent variables. As Székely et al. (2007) point out, if w is integrable and both X and Y have finite variance, then the Taylor expansion of the underlying characteristic functions show, that

$$\lim_{\epsilon\to 0} \frac{\mathcal{V}(\epsilon X, \epsilon Y, w)}{\sqrt{\mathcal{V}^2(\epsilon X, \epsilon X, w)\mathcal{V}^2(\epsilon Y, \epsilon Y, w)}} = \mathrm{Cor}(X,Y) \tag{3.2.4}$$

Thus for uncorrelated X and Y and integrable w the $\mathcal{R}_w$ can be arbitrarily close to zero, even if X and Y are not independent.

One (but not the only) possible weight function is suggested by Székely et al. (2007):

$$w_0(t,s) = \frac{1}{c_p c_q |t|_p^{1+p} |s|_q^{1+q}},$$

where

$$c_p = \frac{\pi^{(1+p)/2}}{\Gamma((1+p)/2)}.$$

Now we have all the required pieces to define the distance covariance:

Definition 3.2.1. (Distance Covariance): Let $X \in \mathcal{X} \subset \mathbb{R}^p$ and $Y \in \mathcal{Y} \subset \mathbb{R}^q$ be two random variables with finite first moments, then the *Distance Covariance* (dCov) between random vectors X and Y is the non-negative number $\mathcal{V}(X,Y)$ defined by

$$\begin{aligned} \mathcal{V}^2(X,Y) &= \mathcal{V}^2(X,Y,w_0) = \|\varphi_{X,Y}(t,s) - \varphi_X(t)\varphi_Y(s)\|_{w_0}^2 \\ &= \frac{1}{c_p c_q} \int_{\mathbb{R}^{p+q}} \frac{|\varphi_{X,Y}(t,s) - \varphi_X(t)\varphi_Y(s)|^2}{|t|_p^{1+p} |s|_q^{1+q}} dtds. \end{aligned} \tag{3.2.5}$$

Similarly we define the *Distance Variance* (or dVar) for a random variable $X \in \mathcal{X} \subset \mathbb{R}^n$ as $\mathcal{V}^2(X) = \mathcal{V}^2(X,X,w_0) = \|f_{X,Y}(t,s) - f_X(t)f_Y(s)\|_{w_0}^2$ and finally the *Distance Correlation* (or dCor) as $\mathcal{R}$, where

$$\mathcal{R}^2(X,Y) = \begin{cases} \frac{\mathcal{V}^2(X,Y)}{\sqrt{\mathcal{V}^2(X)\mathcal{V}^2(Y)}}, & \mathcal{V}^2(X)\mathcal{V}^2(Y) > 0, \\ 0, & \mathcal{V}^2(X)\mathcal{V}^2(Y) = 0. \end{cases}$$

So it follows that for any two non-constant random variables X and Y the distance correlation $\mathcal{R}(X,Y)$ is equal to zero if and only if $f_{X,Y} = f_X f_Y$, i.e. the random variables X and Y are independent. This is a theoretical result and $\mathcal{R}(X,Y)$ would be identically zero only under an infinite sample. In practice we have a finite sample and therefore we are interested in an empirical estimate of $\mathcal{R}(X,Y)$. We discuss this in the following section.

3.2.4 Empirical Estimate of dCor

So far we have provided the theoretical definition of the distance correlation and how it can be used to measure the dependence between two random variables. Now we provide the empirical estimate of the distance correlation from a finite sample Székely et al. (2007). The connection between the theoretical definition and empirical estimate will be made in Section 3.2.5.

Let $X \in \mathcal{X} \subset \mathbb{R}^p$ and $Y \in \mathcal{Y} \subset \mathbb{R}^q$ be two random variables and suppose we have n observations: $(x, y) = ((x_1, y_1), ..., (x_n, y_n))$. Then we define the *Gram distance matrices* K and L as $K_{ij} = \|x_i - x_j\|$ and $L_{ij} = \|y_i - y_j\|$. Next we define the following auxiliary variables

$$\begin{aligned} K_{i*} &= \frac{1}{n}\sum_{j=1}^{n} K_{ij} \quad K_{*j} = \frac{1}{n}\sum_{i=1}^{n} K_{ij} \qquad K_{**} = \frac{1}{n^2}\sum_{i,j=1}^{n} K_{ij} \\ L_{i*} &= \frac{1}{n}\sum_{j=1}^{n} L_{ij} \quad L_{*j} = \frac{1}{n}\sum_{i=1}^{n} L_{ij} \qquad L_{**} = \frac{1}{n^2}\sum_{i,j=1}^{n} L_{ij} \end{aligned} \tag{3.2.6}$$

Then we define the *centred Gram distance matrices* $\tilde{K}$ and $\tilde{L}$ as

$$\begin{aligned} \tilde{K}_{ij} &= K_{ij} - K_{i*} - K_{*j} + K_{**} \\ \tilde{L}_{ij} &= L_{ij} - L_{i*} - L_{*j} + L_{**} \end{aligned} \tag{3.2.7}$$

Alternatively we can write $\tilde{K} = HKH$ and $\tilde{L} = HLH$, where $H_{ij} = \delta_{ij} - \frac{1}{n}$.
Lets consider what the centred Gram distance matrices $\tilde{K}$ represent. Lets go back to the interpretation of Section 3.2.2. Instead of working with vectors x and y we chose to look at the vectors d_x and d_y of the distances between the points of x and y respectively. The correlation $\mathrm{Cor}(d_x, d_y)$ is the same as $\mathrm{Cor}\left((d_x - \bar{d_x}), (d_y - \bar{d_y})\right)$. Now $d_x - \bar{d_x}$ is the same vector as $K_{ij} - K_{**}$ (if we transform the matrix into a vector and order elements in the correct way), i.e. we normalize by subtracting the mean distance from each element of the vector. On the other hand, to produce $\tilde{K}_{ij}$ from K_{ij}, we normalized by subtracting the mean distance from each of the points x_i and x_j and adding the mean distance. So we can think about $\tilde{K}$ as a vector of distances with a slightly different normalization - one that stronger penalizes the outliers.

Example: In this example we demonstrate the effect on the outliers by $\tilde{K}$ and $K_{ij} - K_{**}$. Consider an example with two outliers: $x = (-1, \underbrace{0, ..., 0}_{98 \text{ times}}, 1)$.
Then $d_x = K = (2, 2, \underbrace{1, ..., 1}_{392 \text{ times}}, \underbrace{0, ..., 0}_{9606 \text{ times}})$ (not considering the ordering of the elements).
Now $d_x - \bar{d_x} = K - K_{**} = (1.96, 1.96, \underbrace{0.96, ..., 0.96}_{392 \text{ times}}, \underbrace{-0.04, ..., -0.04}_{9606 \text{ times}})$, while
$\tilde{K} = (-1.96, -1.96, 0.04, 0.04, \underbrace{0.02, ..., 0.02}_{392 \text{ times}}, \underbrace{-0.004, ..., -0.004}_{9604 \text{ times}})$. We can generalize this: suppose we have n data points and there are n_{out} outliers. Then using $d_x - \bar{d_x}$ will produce $\approx n \times n_{out}$ outliers (this is because every distance involving the outlier as

one of the two points will be an outlier in its own right) while $\tilde{K}$ will keep the same number $\approx n_{out}$ of the outliers.

We provide the definitions for the *empirical estimates* of the distance covariance, variance and correlation:

Definition 3.2.2. (Empirical Distance Covariance): For a finite sample $(x, y) = ((x_1, y_1), ..., (x_n, y_n))$ from the pair of random variables (X, Y) the *Empirical Estimate of the Distance Covariance* $\mathcal{V}_n$ is defined by

$$\mathcal{V}_n^2(x, y) = \frac{1}{n^2} \sum_{i,j=1}^{n} \tilde{K}_{ij} \tilde{L}_{ij} \tag{3.2.8}$$

similarly the *Empirical Estimate of the Distance Variance* $\mathcal{V}_n$ is defined by

$$\mathcal{V}_n^2(x) = \mathcal{V}_n^2(x, x) = \frac{1}{n^2} \sum_{i,j=1}^{n} \tilde{K}_{ij}^2 \tag{3.2.9}$$

finally the *Empirical Estimate of the Distance Correlation* $\mathcal{R}_n$ is defined by

$$\mathcal{R}_n^2(x, y) = \begin{cases} \frac{\mathcal{V}_n^2(x,y)}{\sqrt{\mathcal{V}_n^2(x)\mathcal{V}_n^2(y)}}, & \mathcal{V}_n^2(x)\mathcal{V}_n^2(y) > 0, \\ 0, & \mathcal{V}_n^2(x)\mathcal{V}_n^2(y) = 0. \end{cases} \tag{3.2.10}$$

We should note that there is no inherent difference whether we think about the K as a matrix or a vector of distances. If we choose to think about K as a vector rather than a matrix we can think of the empirical estimate of distance covariance as a dot product between $\tilde{K}$ and $\tilde{L}$ (and normalizing by dividing by the its length, recall that $\|K\| = \|L\| = n^2$). There is another interpretation of $\mathcal{V}_n^2$:

Proposition 3.2.1. The empirical estimate of the distance covariance $\mathcal{V}_n^2$ can be written as

$$\mathcal{V}_n^2(x, y) = \frac{1}{n^2} \operatorname{Tr} \tilde{K}\tilde{L} \tag{3.2.11}$$

Proof. Matrices $\tilde{K}$ and $\tilde{L}$ are symmetric, therefore $\tilde{L} = \tilde{L}^T$ and so

$$\sum_{i,j=1}^{n} \tilde{K}_{ij} \tilde{L}_{ij} = \sum_{i,j=1}^{n} \tilde{K}_{ij} \tilde{L}_{ji} = \operatorname{Tr} \tilde{K}\tilde{L} \tag{3.2.12}$$

□

This is of interest because the trace of a matrix is equal to the sum of its eigenvalues, i.e.

$$\operatorname{Tr} A = \sum_{i=1}^{n} \lambda_i \tag{3.2.13}$$

where λ_i's are the eigenvalues of A. So the empirical estimate of the distance covariance is equal to (normalized) sum of all the eigenvalues of the product of the centred Gram distance matrices $\tilde{K}\tilde{L}$. This gives a different interpretation of the meaning of this independence criteria.

3.2.5 Theoretical Justification of dCor

So far we have given the theoretical definition of the distance correlation and its empirical estimate, the last step that has to be made is to show that the empirical estimate is indeed meaningful. To do so we provide a theorem and its corollary taken from Székely et al. (2007), which show that the empirical estimates of distance covariance and correlation tend to the real distance covariance and correlation as the number of observations ten to infinity. For the proofs please refer to the original paper Theorem 2 and Corollary 1.

Theorem 3.2.2 (Székely et al. (2007))**.** If $\mathrm{E}|X|_p < \infty$ and $\mathrm{E}|Y|_q < \infty$, then almost surely

$$\lim_{n\to\infty} \mathcal{V}_n^2(x, y) = \mathcal{V}^2(X, Y) \tag{3.2.14}$$

Corollary 3.2.1 (Székely et al. (2007))**.** If $\mathrm{E}(|X|_p + |Y|_q) < \infty$, then almost surely

$$\lim_{n\to\infty} \mathcal{R}_n^2(x, y) = \mathcal{R}^2(X, Y) \tag{3.2.15}$$

This theorem finishes the presentation of the first of the independence criteria, the correlation of distances or dCor. We will come back to it in later sections when we will be discussing statistical tests to evaluate the probability of random variables being independent.

3.3 Kernel Canonical Correlation

3.3.1 Introduction

In this section we discuss the second independence criteria, the kernel canonical correlation (or $\mathcal{F}$-correlation) first introduced by Bach and Jordan (2002). The

intuition behind their criteria is very different from the previous one, the correlation of distances, but as we will see later, the empirical estimate has many similarities. We follow the same pattern as before: we start by the motivation why the canonical correlation is worth looking at, and continue with a formal definition and empirical estimate.

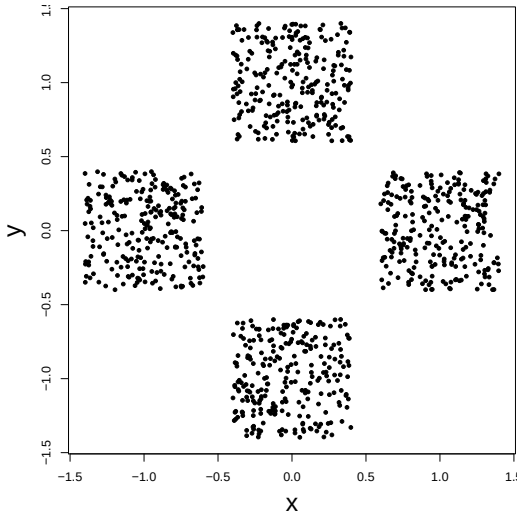
3.3.2 Motivation

The underlying idea behind the correlation of distances was: "correlation catches only the linear trends, can we upgrade it to catch any sort of relationship?" This time we will be asking a slightly different question: "can I apply some simple transformation to data to extract a clear signal?" This is not a well defined question, we need to be more specific about what we mean by a "simple" transformation and a "clear" signal. For the time being lets define the simple transformation as a continuous function on the data and clear signal as a linear relationship. Then the previously very vague question may be written as

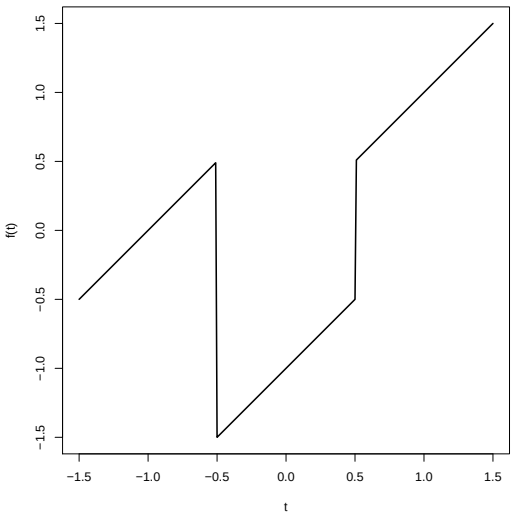
$$\rho_{\mathcal{F}} = \max_{(f,g)\in\mathcal{F}\times\mathcal{G}} \mathrm{Cor}(f(x), g(y)), \tag{3.3.1}$$

Here $x \in \mathcal{X} \subseteq \mathbb{R}^p$ and $y \in \mathcal{Y} \subseteq \mathbb{R}^q$ are two random variables and $\mathcal{F}$ is a space of all continuous functions from $\mathcal{X}$ to $\mathbb{R}$ while $\mathcal{G}$ is a space of all continuous functions from $\mathcal{Y}$ to $\mathbb{R}$.

Example: To illustrate this idea consider a simple example. Consider the data from Figure 3.4a. The variables x and y are dependent, but have zero correlation. In this case we may extract a clear linear signal by applying a simple piece-wise linear function shown in Figure 3.4b (it is essentially a permutation function that moves everything from the interval $(-1.5, -0.5)$ to the interval $(-0.5, 0.5)$ and vice versa). After applying this function to the variable y only we don't achieve anything (see Figure 3.5a) but if we apply f to both variables we get a very clear signal, correlation of -0.73. In this example we used the same function for both variables, but in general they would be different. Also we do not claim that f is the function that maximizes $\mathrm{Cor}(f(x), g(y))$, it simply is some function that extracts a clear signal and that is sufficient to show that x and y are not independent.

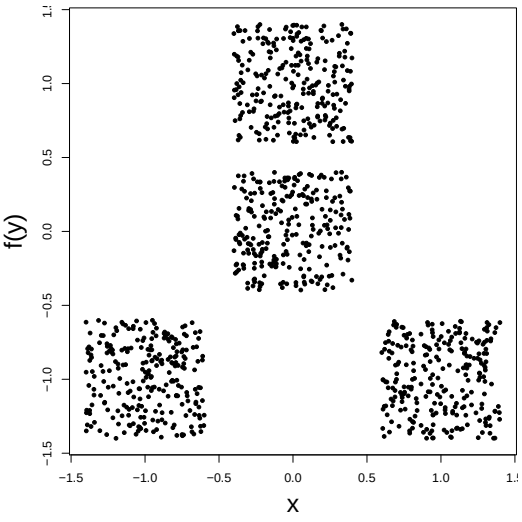


(a) Data that is dependent but has zero correlation, $\mathrm{Cor}(x, y) = 0.01$.

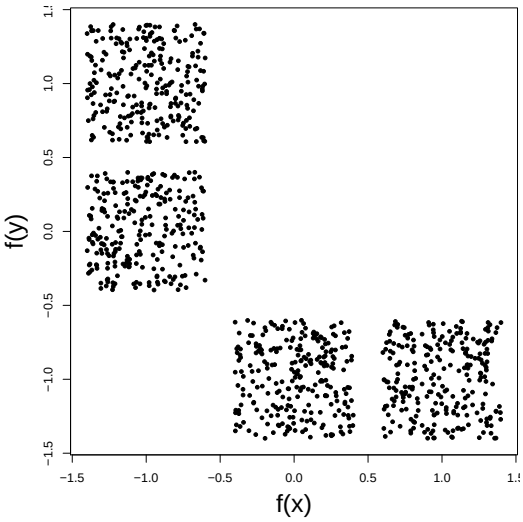


(b) Example of a function f that helps to extract a linear dependency from the data

Fig. 3.4 Example of data that is dependent but has zero correlation and a function that allows us to extract a linear dependence from this data.



(a) Data with function f applied only to y, still zero correlation, $\mathrm{Cor}(x, f(y)) = 0$.



(b) Data with function f applied to both x and y, $\mathrm{Cor}(f(x), f(y)) = -0.73$.

Fig. 3.5 Data from Figure 3.4a with function f from the Figure 3.4b applied to one and both variables.

3.3.3 Reproducing Kernel Hilbert Space

After some motivation why the transformations in Equation 3.3.1 are potentially helpful to discover dependencies we provide a formal definition. The definition of

the $\mathcal{F}$-correlation relies heavily on the Hilbert spaces theory. To make our work self-contained we provide the necessary theory in this section. We start by defining normed and Hilbert spaces (definitions mainly taken from Sutherland (2009)), then we introduce the reproducing kernels and the Reproducing Kernel Hilbert Spaces as defined in Gretton et al. (2005) and Gretton et al. (2008).

Definition 3.3.1. (Metric space): Let $\mathcal{X}$ be an arbitrary non-empty set. Then a function $d : X \times X \to \mathbb{R}$ is a *metric*, or *distance function*, on X if

1. $d(x, y) \geq 0$, $\forall x, y \in \mathcal{X}$, and $d(x, y) = 0$ iff $x = y$;
2. $d(x, y) = d(y, x)$, $\forall x, y \in \mathcal{X}$;
3. $d(x, y) \leq d(x, z) + d(z, y)$, $\forall x, y, z \in \mathcal{X}$.

A *metric space* $(\mathcal{X}, d(\cdot, \cdot))$ is a set $\mathcal{X}$ equipped with a metric d.

Definition 3.3.2. (Normed space): Let $\mathcal{F}$ be a vector space over $\mathbb{R}$. A function $\|\cdot\|_{\mathcal{F}} : \mathcal{F} \to [0, \infty)$ is said to be a *norm* on $\mathcal{F}$ if

1. $\|f\|_{\mathcal{F}} = 0$ iff $f = 0$ (norm separates points);
2. $\|\alpha f\|_{\mathcal{F}} = |\alpha| \|f\|_{\mathcal{F}}$, $\forall \alpha \in \mathbb{R}$, $\forall f \in \mathcal{F}$ (positive homogeneity);
3. $\|f + g\|_{\mathcal{F}} \leq \|f\|_{\mathcal{F}} + \|g\|_{\mathcal{F}}$, $\forall f, g \in \mathcal{F}$ (triangle inequality).

A *normed vector space* $(\mathcal{F}, \|\cdot\|_{\mathcal{F}})$ is a vector space $\mathcal{F}$ equipped with a norm $\|\cdot\|_{\mathcal{F}}$.

Definition 3.3.3. (Convergent sequence): A sequence $\{f_n\}_{n=1}^{\infty}$ of elements of a normed vector space $(\mathcal{F}, \|\cdot\|_{\mathcal{F}})$ is said *to converge* to $f \in \mathcal{F}$ if for every $\epsilon > 0$, there exists $N \in \mathbb{N}$, such that for all $n \geq N$, $\|f_n - f\| < \epsilon$.

Definition 3.3.4. (Cauchy sequence): A sequence $\{f_n\}_{n=1}^{\infty}$ of elements of a normed vector space $(\mathcal{F}, \|\cdot\|_{\mathcal{F}})$ is said to be a *Cauchy sequence* if for every $\epsilon > 0$, there exists $N \in \mathbb{N}$, such that for all $n, m \geq N$, $\|f_n - f_m\| < \epsilon$.

Definition 3.3.5. (Complete space): A metric space $\mathcal{X}$ is *complete*, if every Cauchy sequence in $\mathcal{X}$ converges (to a point of $\mathcal{X}$).

Definition 3.3.6. (Banach space): *Banach space* is a complete normed space, i.e. it contains the limits of all its Cauchy sequences.

Definition 3.3.7. (Inner product space): Let $\mathcal{F}$ be a vector space over $\mathbb{R}$. A function $\langle \cdot, \cdot \rangle_{\mathcal{F}} : \mathcal{F} \times \mathcal{F} \to \mathbb{R}$ is said to be *an inner product* on $\mathcal{F}$ if

1. $\langle \alpha f + \beta g, h \rangle_{\mathcal{F}} = \langle \alpha f, h \rangle_{\mathcal{F}} + \langle \beta g, h \rangle_{\mathcal{F}}$, $\forall \alpha, \beta \in \mathbb{R}$, $\forall f, g, h \in \mathcal{F}$;
2. $\langle f, g \cdot \rangle_{\mathcal{F}} = \langle g, f \cdot \rangle_{\mathcal{F}}$, $\forall f, g \in \mathcal{F}$;
3. $\langle f, f \cdot \rangle_{\mathcal{F}} \geq 0$ and $\langle f, f \cdot \rangle_{\mathcal{F}} = 0$ iff $f = 0$.

An *inner product space* $(\mathcal{F}, \langle \cdot, \cdot \rangle_{\mathcal{F}})$ is a vector space $\mathcal{F}$ equipped with an inner product $\langle \cdot, \cdot \rangle_{\mathcal{F}}$.

Definition 3.3.8. (Hilbert space): *Hilbert space* is a complete inner product space.

Definition 3.3.9. (Linear operator): Let $\mathcal{F}$ and $\mathcal{G}$ be normed linear spaces over $\mathbb{R}$, then an operator $L : \mathcal{F} \to \mathcal{G}$, is called a *linear* operator iff

1. $L(\alpha f) = \alpha L f$, $\forall \alpha \in \mathbb{R}$, $\forall f \in \mathcal{F}$ (homogeneity);
2. $L(f + g) = Lf + Lg$, $\forall f, g \in \mathcal{F}$ (additivity).

Definition 3.3.10. (Operator norm): The *operator norm* of a linear operator $L : \mathcal{F} \to \mathcal{G}$ is defined as

$$\|L\| = \sup_{f \in \mathcal{F}} \frac{\|Lf\|_{\mathcal{G}}}{\|f\|_{\mathcal{F}}}$$

Definition 3.3.11. (Bounded operator): The linear operator $L : \mathcal{F} \to \mathcal{G}$ is said to be *bounded operator* if $\|L\| < \infty$.

Definition 3.3.12. (Positive definite kernel): Let $\mathcal{X}$ be an arbitrary non-empty set. Let $\mathcal{F}$ be a Hilbert space of functions $f : \mathcal{X} \to \mathbb{R}$. A asymmetric function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is called a *positive definite kernel* of $\mathcal{F}$ if

$$\sum_{i,j=1}^{n} c_i c_j k(x_i, x_j) \geq 0 \tag{3.3.2}$$

for all $n \in \mathbb{N}$, $x_1, ..., x_n \in \mathcal{X}$ and $c_1, ..., c_n \in \mathbb{R}$.

Example An example of a positive definite kernel is the widely used Gaussian (sometimes called radial basis function) kernel:

$$k(x, x') = \exp\left(\frac{-(x - x')^2}{2\sigma^2}\right) \tag{3.3.3}$$

Definition 3.3.13. (Reproducing kernel): Let $\mathcal{X}$ be an arbitrary non-empty set. Let $\mathcal{F}$ be a Hilbert space of functions $f : \mathcal{X} \to \mathbb{R}$. A symmetric function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is called a *reproducing kernel* of $\mathcal{F}$ if

1. $\forall x \in \mathcal{X}$, $k(x, \cdot) \in \mathcal{F}$;

2. $\forall x \in \mathcal{X}, \forall f \in \mathcal{F}$, $\langle f, k(x, \cdot)\rangle_{\mathcal{F}} = f(x)$ (the reproducing property).

Let $\mathcal{X}$ be an arbitrary non-empty set. Let $\mathcal{F}$ be a Hilbert space of functions $f : \mathcal{X} \to \mathbb{R}$. Then a *feature map* is a function $\phi : \mathcal{X} \to \mathcal{F}$. For every feature map ϕ there is a corresponding reproducing kernel k such that

$$k(x, x') = \langle \phi_x(\cdot), \phi_{x'}(\cdot)\rangle_{\mathcal{F}} \tag{3.3.4}$$

Choose a feature map to be $\phi_x(\cdot) = k(x, \cdot)$, the it follows directly from Definition 3.3.13 that

$$\langle \phi_x(\cdot), \phi_{x'}(\cdot)\rangle_{\mathcal{F}} = \langle k(x, \cdot), k(x', \cdot)\rangle_{\mathcal{F}} = k(x, x') \tag{3.3.5}$$

Example To illustrate the relationship between the feature maps and the reproducing kernels we go back to our example of the Gaussian kernel from Equation 3.3.3. In this case we can show explicitly that for the corresponding feature map

$$\phi_x(y) = (\pi\sigma^2/2)^{-1/4} \exp\left(\frac{-(x-y)^2}{\sigma^2}\right) \tag{3.3.6}$$

with an inner product space $\mathcal{F}$ equipped with an inner product

$$\langle f(\cdot), g(\cdot)\rangle_{\mathcal{F}} = \int_{-\infty}^{\infty} f(y)g(y)dy \tag{3.3.7}$$

relationship $k(x, x') = \langle \phi_x(\cdot), \phi_{x'}(\cdot)\rangle_{\mathcal{F}}$ holds.

$$
\begin{aligned}
\langle \phi_x(\cdot), \phi_{x'}(\cdot) \rangle_{\mathcal{F}} &= \int_{-\infty}^{\infty} \phi_x(y)\phi_{x'}(y)dy \\
&= \frac{1}{\sqrt{\pi\sigma^2/2}} \int_{-\infty}^{\infty} \exp\left(\frac{-(x-y)^2}{\sigma^2}\right) \exp\left(\frac{-(x'-y)^2}{\sigma^2}\right) dy \\
&= \frac{1}{\sqrt{\pi\sigma^2/2}} \int_{-\infty}^{\infty} \exp\left(\frac{-(x^2 - 2xy + y^2 + x'^2 - 2x'y + y^2)}{\sigma^2}\right) dy \\
&= \frac{1}{\sqrt{\pi\sigma^2/2}} \int_{-\infty}^{\infty} \exp\left(\frac{-\left(2y^2 - 4y\frac{x+x'}{2} + 2\left(\frac{x+x'}{2}\right)^2 - 2\left(\frac{x+x'}{2}\right)^2 + x^2 + x'^2\right)}{\sigma^2}\right) dy \\
&= \frac{1}{\sqrt{\pi\sigma^2/2}} \int_{-\infty}^{\infty} \exp\left(\frac{-2\left(y - \frac{x+x'}{2}\right)^2 - \frac{1}{2}(x-x')^2}{\sigma^2}\right) dy \\
&= \exp\left(\frac{-(x-x')^2}{2\sigma^2}\right) \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi(\sigma/2)^2}} \exp\left(\frac{-\left(y - \frac{x+x'}{2}\right)^2}{2(\sigma/2)^2}\right) dy \\
&= \exp\left(\frac{-(x-x')^2}{2\sigma^2}\right) = k(x, x')
\end{aligned}
\tag{3.3.8}
$$

Definition 3.3.14. Let $\mathcal{X}$ be an arbitrary non-empty set. Let $\mathcal{F}$ be a Hilbert space of functions $f : \mathcal{X} \to \mathbb{R}$. Given a feature map $\phi : \mathcal{X} \to \mathbb{R}$ and a finite sample $x = \{x_1, ..., x_n\} \in X$, the *$\phi$-image* of x in $\mathcal{F}$, denoted $\mathcal{F}_{\phi_x}$, is a linear subspace of $\mathcal{F}$ spanned by the functions $\phi_{x_i}(\cdot)$, for all $x_i \in x$, i.e.

$$
\mathcal{F}_\phi = \left(\sum_{i=1}^{n} \alpha_i \phi_{x_i}(\cdot) \colon \forall \alpha_i \in \mathbb{R} \text{ and } x_i \in x \right) \tag{3.3.9}
$$

Definition 3.3.15. (Reproducing Kernel Hilbert Space): Let $\mathcal{X}$ be an arbitrary non-empty set and let $\mathcal{F}$ be a Hilbert space of functions $f : \mathcal{X} \to \mathbb{R}$. For any fixed $x \in \mathcal{X}$ let $\delta_x : \mathcal{F} \to \mathbb{R}$ be defined as $\delta_x : f \to f(x)$. Then the Hilbert space $\mathcal{F}$ is said to be a *Reproducing Kernel Hilbert Space* (RKHS) if δ_x is a bounded operator $\forall x \in \mathcal{X}$.

The following definitions are taken from Hunter and Nachtergaele (2001).

Definition 3.3.16. If x, y are vectors in a Hilbert space $\mathcal{H}$, then we say that x and y are *orthogonal*, written $x \perp y$, if $\langle x, y \rangle = 0$. We say that subsets A and B are orthogonal, written $A \perp B$, if $\forall x \in A, y \in B$ $x \perp y$ holds. The *orthogonal complement*

$A^{\perp}$ of a subset A is the set of vectors orthogonal to A.

$$A^{\perp} = \{x \in \mathcal{H} \mid x \perp y, \forall y \in A\}$$

Definition 3.3.17. Let $\mathcal{H}$ be a Hilbert space. A maximal orthonormal subset $u \subset \mathcal{H}$ is called an *orthonormal basis* of $\mathcal{H}$.

Lemma 3.3.1 (Hunter and Nachtergaele (2001))**.** A Hilbert space $\mathcal{H}$ is separable iff $\mathcal{H}$ has a countable orthonormal basis $u \subset \mathcal{H}$.

3.3.4 Definition of the $\mathcal{F}$-correlation

After providing the necessary definitions we introduce the $\mathcal{F}$-correlation

Definition 3.3.18. ($\mathcal{F}$-correlation): Let $X \in \mathcal{X} \subset \mathbb{R}^p$ and $Y \in \mathcal{Y} \subset \mathbb{R}^q$ be two random variables. Let $\mathcal{F}$ and $\mathcal{G}$ be the Hilbert spaces of functions from $\mathcal{X}$ and $\mathcal{Y}$ respectively to $\mathbb{R}$. Let $\phi : \mathcal{X} \to \mathcal{F}$ and $\psi : \mathcal{Y} \to \mathcal{G}$ be feature maps corresponding to some reproducing kernels. Then the *$\mathcal{F}$-correlation* $\rho_{\mathcal{F}}$ between ϕ_x and ψ_y is defined as

$$\rho_{\mathcal{F}}(X, Y) = \max_{(f,g)\in\mathcal{F}\times\mathcal{G}} \mathrm{Cor}(\langle \phi_X, f\rangle_{\mathcal{F}}, \langle \psi_Y, g\rangle_{\mathcal{G}}),$$

As ϕ_x and ψ_y both correspond to some reproducing kernels it follows from Definition 3.3.13 that $\langle \phi_x, f\rangle_{\mathcal{F}} = f(x), \forall f \in \mathcal{F}$ and $\langle \psi_y, g\rangle_{\mathcal{G}} = g(y), \forall g \in \mathcal{G}$. We can rewrite Definition 3.3.18 in a simpler form

$$\rho_{\mathcal{F}} = \max_{(f,g)\in\mathcal{F}\times\mathcal{G}} \mathrm{Cor}(f(X), g(Y)). \tag{3.3.10}$$

Note that the $\mathcal{F}$-correlation depends on two Hilbert spaces $\mathcal{F}$ and $\mathcal{G}$ which could in principal be different. For the notational convenience we will keep calling it $\mathcal{F}$-correlation.
The following theorem provides the connection between the $\mathcal{F}$-correlation and the general independence (proof is provided in the original paper).

Theorem 3.3.2 (Bach and Jordan (2002))**.** (Independence and $\mathcal{F}$-correlation): If $\mathcal{F}$ and $\mathcal{G}$ are the reproducing kernel Hilbert spaces corresponding to the Gaussian kernels on $\mathcal{X} = \mathcal{Y} = \mathbb{R}$, then $\rho_{\mathcal{F}} = 0$ if and only if the variables X and Y are independent.

After we have motivated the usage of a new independence criterion, namely the $\mathcal{F}$-correlation, and have formally defined it, we show how to estimate it for finite samples in next section.

3.3.5 Empirical Estimate of the $\mathcal{F}$-correlation

So far we have provided the formal definition of the $\mathcal{F}$-correlation and how it can be used to measure the dependence between two random variables. Next we provide the empirical estimate for the $\mathcal{F}$-correlation from a finite sample.
Let $X \in \mathcal{X} \subset \mathbb{R}^p$ and $Y \in \mathcal{Y} \subset \mathbb{R}^q$ be two random variables and suppose we have n observations: $(x, y) = ((x_1, y_1), ..., (x_n, y_n))$. Suppose we want to make an empirical estimate of the $\mathcal{F}$-correlation between the random variables X and Y, i.e. we want to find

$$\widehat{\rho_{\mathcal{F}}}(x, y) = \max_{(f,g)\in\mathcal{F}\times\mathcal{G}} \widehat{\mathrm{Cor}}\Big(f(x), g(y)\Big) \tag{3.3.11}$$

Here and throughout this thesis the hat symbol will mean the empirical estimate, i.e. $\widehat{\rho_{\mathcal{F}}}$ denotes the empirical estimate of $\rho_{\mathcal{F}}$.
Let $\mathcal{F}_{\phi_x} \subseteq \mathcal{F}$ and $\mathcal{G}_{\psi_y} \subseteq \mathcal{G}$ represent the ϕ_x and ψ_y images of the data points. To make the notation less cumbersome we will use the short hand notation $\mathcal{F}_\phi = \mathcal{F}_{\phi_x}$ and $\mathcal{G}_\psi = \mathcal{G}_{\psi_y}$. Then we can write any $f \in \mathcal{F}$ and $g \in \mathcal{G}$ as $f = \sum_{i=1}^n \alpha_i \phi_{x_i} + f^\perp$ and $g = \sum_{i=1}^n \beta_i \psi_{y_i} + g^\perp$, where $f^\perp$ and $g^\perp$ are in the orthogonal complements to $\mathcal{F}_\phi$ and $\mathcal{G}_\psi$ respectively. Now assuming that all the x's and y's are distinct and using the feature maps related to the Gaussian kernels we have that $(\phi_{x_i}, i = 1, ..., n)$ are the basis of $\mathcal{F}_\phi$ (similarly the ψ_{y_i} are the basis for $\mathcal{G}_\psi$). Therefore we can write the empirical estimate of the covariance as

$$\widehat{\mathrm{Cov}}\Big(\langle \phi_x, f\rangle_{\mathcal{F}_\phi}, \langle \psi_y, g\rangle_{\mathcal{G}_\psi}\Big) = \frac{1}{n}\sum_{i=1}^n \langle \phi_{x_i}, f\rangle_{\mathcal{F}_\phi} \langle \psi_{y_i}, g\rangle_{\mathcal{G}_\psi}$$

and rewrite Equation 3.3.11 in a more convenient form

$$\widehat{\rho_{\mathcal{F}}}(x, y) = \max_{(f,g)\in\mathcal{F}_\phi\times\mathcal{G}_\psi} \mathrm{Cor}\Big(f(x), g(y)\Big)$$

The following proposition provides us with a way to transform the problem of maximizing over all functions into a linear maximization problem

Proposition 3.3.3. Let $(x, y) = ((x_1, y_1), ..., (x_n, y_n))$ be observations from the two random variables X and Y. Let K and L, defined as $K_{ij} = k(x_i, x_j)$ and $L_{ij} = l(y_i, y_j)$ for some kernels k and l, be their kernel matrices respectively, then

$$\max_{(f,g)\in\mathcal{F}_\phi\times\mathcal{G}_\psi} \widehat{\mathrm{Cov}}(f(x), g(y)) = \max_{\alpha,\beta\in\mathbb{R}^n} \frac{1}{n}\alpha^T \tilde{K}\tilde{L}\beta \tag{3.3.12}$$

where $\tilde{K} = KH$, $\tilde{L} = LH$ and H is a matrix with elements $H_{ij} = \delta_{ij} - 1/n$.

Proof. Proof is provided in the Appendix, Section C , Proposition C.1.1. □

Proposition 3.3.4. Let M be an $n \times n$ matrix, then

$$\max_{\|\alpha\|=\|\beta\|=1} \alpha^T M \beta = |\lambda_{\max}|$$

where $\lambda_{\max}$ is the largest (in absolute value) eigenvalue of M.

Proof. Let λ_i be the eigenvalues of M and e_i be the corresponding eigenvectors. We use the eigenvalue and eigenvector decomposition for M, i.e.

$$M = \Sigma^T D \Sigma$$

where D is a diagonal matrix with elements $D_{ii} = \lambda_i$ and Σ is a matrix of eigenvectors of M with i^th column being eigenvector e_i, then we can rewrite the maximization problem as

$$\begin{aligned}
\max_{\|\alpha\|=\|\beta\|=1} \alpha^T M \beta &= \max_{\|\alpha\|=\|\beta\|=1} \alpha^T \Sigma^T D \Sigma \beta \\
&= \max_{\|\alpha\|=\|\beta\|=1} (\Sigma\alpha)^T D (\Sigma\beta) \\
&= \max_{\|\alpha'\|=\|\beta'\|=1} \alpha'^T D \beta' \\
&= \max_{\|\alpha'\|=\|\beta'\|=1} \sum_{i=1}^{n} \lambda_i \alpha'_i \beta'_i = |\lambda_{\max}|
\end{aligned}$$

□

Note: It is important to note that if we are considering an optimisation problem with a constraint $\|\alpha\| = \|\beta\| = 1$, then it follows from Proposition 3.3.4 that

$$\max_{\|\alpha\|=\|\beta\|=1} \alpha^T \tilde{K} \tilde{L} \beta = |\lambda_{\max}| \tag{3.3.13}$$

where $\lambda_{\max}$ is the largest (in absolute value) eigenvalue of the matrix $\tilde{K}\tilde{L}$. So the problem of maximizing over the space of functions $\mathcal{F}$ becomes a problem of finding the largest eigenvalue. Another interesting observation is that in the case of the $\mathcal{F}$-correlation we are looking for the largest eigenvalue of the product of the kernel matrices, while in the case of the distance covariance we were looking for the sum of all eigenvalues of the product of the distance matrices. In both cases we have independence if the corresponding values (the maximal eigenvalue or the sum of the eigenvalues) are equal to zero. Now if the largest eigenvalue is zero, then the sum of eigenvalues will also be zero, equivalently if the sum of all eigenvalues is zero, then the largest eigenvalue

must be zero as well. Note that here we are dealing with only non-negative eigenvalues therefore "largest" eigenvalue is equivalent to "largest in the absolute value" (this is the case because $\tilde{K}$ and $\tilde{L}$ are symmetric positive definite matrices and therefore there product $\tilde{K}\tilde{L}$ is similar to a positive semi-definite matrix $\tilde{K}^{-1/2}(\tilde{K}\tilde{L})\tilde{K}^{1/2} = \tilde{K}^{1/2}\tilde{L}\tilde{K}^{1/2}$ which is congruent to $\tilde{L}$).
After similar calculations we also obtain:

$$\widehat{\mathrm{Var}}(f(x)) = \frac{1}{m}\alpha^T\tilde{K}\tilde{K}\alpha$$

and

$$\widehat{\mathrm{Var}}(g(y)) = \frac{1}{m}\beta^T\tilde{L}\tilde{L}\beta.$$

Putting these results together, the empirical estimate of $\mathcal{F}$-correlation becomes that of performing the following maximization:

$$\hat{\rho}_{\mathcal{F}}(x,y) = \max_{\alpha,\beta\in\mathbb{R}^N} \frac{\alpha^T\tilde{K}\tilde{L}\beta}{\left(\alpha^T\tilde{K}\tilde{K}\alpha\right)^{1/2}\left(\beta^T\tilde{L}\tilde{L}\beta\right)^{1/2}} \tag{3.3.14}$$

Without the loss of generality we may use the normalization $\alpha^T\tilde{K}\tilde{K}\alpha = 1$ and $\beta^T\tilde{L}\tilde{L}\beta = 1$.
So finding the empirical estimate of $\rho_{\mathcal{F}}$ becomes a constrained maximization problem:

$$\begin{aligned} &\max\ \alpha^T\tilde{K}\tilde{L}\beta \\ &\text{w.r.t. } \alpha^T\tilde{K}\tilde{K}\alpha = \beta^T\tilde{L}\tilde{L}\beta = 1 \end{aligned} \tag{3.3.15}$$

Compare this with the constrained optimization problem for the $\mathcal{F}$-covariance we have discussed in Proposition 3.3.3

$$\begin{aligned} &\max\ \alpha^T\tilde{K}\tilde{L}\beta \\ &\text{w.r.t. } \alpha^T\alpha = \beta^T\beta = 1 \end{aligned} \tag{3.3.16}$$

It is the same optimization problem, only the constraints are different.
To solve this constrained optimization problem we use Lagrange multipliers. Differentiating 3.3.14 with respect to α and β yields:

$$\begin{cases} \tilde{K}\tilde{L}\beta &= \frac{\alpha^T\tilde{K}\tilde{L}\beta}{\alpha^T\tilde{K}\tilde{K}\alpha}\tilde{K}\tilde{K}\alpha \\ \tilde{L}\tilde{K}\alpha &= \frac{\alpha^T\tilde{K}\tilde{L}\beta}{\beta^T\tilde{L}\tilde{L}\beta}\tilde{L}\tilde{L}\beta \end{cases} \tag{3.3.17}$$

Using the normalization conditions $\alpha^T\tilde{K}\tilde{K}\alpha = \beta^T\tilde{L}\tilde{L}\beta = 1$ we can rewrite this as

$$\begin{pmatrix} 0 & \tilde{K}\tilde{L} \\ \tilde{L}\tilde{K} & 0 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \left(\alpha^T \tilde{K}\tilde{L}\beta\right) \begin{pmatrix} \tilde{K}\tilde{K} & 0 \\ 0 & \tilde{L}\tilde{L} \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \tag{3.3.18}$$

Naming $\alpha^T \tilde{K}\tilde{L}\beta = \lambda$ (recall that this is the quantity we want to maximize over) we can write the optimization problem in Equation 3.3.14 as a generalized eigenvalue problem

$$\begin{pmatrix} 0 & \tilde{K}\tilde{L} \\ \tilde{K}\tilde{L} & 0 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \lambda \begin{pmatrix} \tilde{K}\tilde{K} & 0 \\ 0 & \tilde{L}\tilde{L} \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \tag{3.3.19}$$

Or alternatively:

$$\begin{pmatrix} \tilde{K}\tilde{K} & \tilde{K}\tilde{L} \\ \tilde{L}\tilde{K} & \tilde{L}\tilde{L} \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = (1+\lambda) \begin{pmatrix} \tilde{K}\tilde{K} & 0 \\ 0 & \tilde{L}\tilde{L} \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \tag{3.3.20}$$

The generalized eigenvalue problem in 3.3.19 has eigenvalues $\{\lambda_1, -\lambda_1, ..., \lambda_p, -\lambda_p\}$. If λ' is an eigenvalue with a corresponding eigenvector $(\alpha'\ \beta')^T$ then $-\lambda'$ is also an eigenvalue with a corresponding eigenvector $(\alpha'\ -\beta')^T$. Alternative version 3.3.20 has eigenvalues $\{1+\lambda_1, 1-\lambda_1, ..., 1+\lambda_p, 1-\lambda_p\}$. We note, that for 3.3.20 finding the maximal generalized eigenvalue $1+\lambda_{max}$, where λ_{max} is the empirical estimate of the $\mathcal{F}$-correlation, is equivalent to finding the minimal generalized eigenvalue, $1-\lambda_{max}$. So we can find the canonical correlation by finding the minimal eigenvalue of 3.3.20. Bach and Jordan (2002) points out that the un-regularised $\mathcal{F}$-correlation is not a particularly useful estimate. In many cases (for example the Gaussian kernels and distinct data points) the kernel matrices are invertible and the $\mathcal{F}$-correlation is identically equal to one. To overcome this issue they introduced the *Regularised $\mathcal{F}$-correlation*. In the regularized approach we replace the generalized eigenvalue problem from Equation 3.3.19, by

$$\begin{pmatrix} 0 & \tilde{K}\tilde{L} \\ \tilde{L}\tilde{K} & 0 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \lambda \begin{pmatrix} \left(\tilde{K} + \frac{p\kappa}{2} I\right)^2 & 0 \\ 0 & \left(\tilde{L} + \frac{q\kappa}{2} I\right)^2 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \tag{3.3.21}$$

here κ is the regularization parameter and I is the identity matrix. For the more detailed explanation off this regularization approach please refer to the Appendix C.

3.3.6 Kernel Generalized Variance

So far we have discussed the kernel canonical correlation or the $\mathcal{F}$-correlation that was introduced in the Bach and Jordan (2002). Finding an empirical estimate of the

$\mathcal{F}$-correlation is equivalent to finding the largest eigenvalue of the generalized eigenvalue problem 3.3.19.
Bach and Jordan (2002) proposed the generalization of $\mathcal{F}$-correlation - the Kernel Generalized Variance (KGV). KGV uses all the eigenvalues of the generalised eigenvalue problem 3.3.20 (in this aspect it is more similar to the distance covariance), KGV also has a close relationship with the mutual information.
We start by defining the mutual information and showing its relationship to the eigenvalues in the Gaussian variable case.

Definition 3.3.19. The *mutual information* of two random variables X and Y is defined as

$$I(X,Y) = \int_X \int_Y p(x,y) \log \left(\frac{p(x,y)}{p(x)p(y)} \right) dxdy \tag{3.3.22}$$

where $p(x,y)$ is the joint probability distribution function of X and Y, and $p(x)$ and $p(y)$ are the marginal probability distribution functions of X and Y respectively.

Proposition 3.3.5. Two random variables X and Y are independent if and only if their mutual information $I(X,Y)$ is equal to zero.

Proof. Proof is provided in the Appendix C, Proposition C.1.3. □

For Gaussian random variables there is a simple relationship between $\mathcal{F}$-correlation and the mutual information.

Proposition 3.3.6. Let $X \subset \mathbb{R}^p$ and $Y \subset \mathbb{R}^q$ be two multivariate Gaussian random variables with covariance matrix $C = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix}$. Then their mutual information is

$$I(X,Y) = -\frac{1}{2} \log \left(\frac{\det C}{\det C_{11} \det C_{22}} \right) \tag{3.3.23}$$

Proof. Proof is provided in the Appendix C, Proposition C.1.4. □

Recall from Section 3.3.5, that for two multivariate Gaussian random variable their $\mathcal{F}$-correlation is equal to the largest eigenvalue λ_{max} of the generalized eigenvalue problem

$$\begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = (1+\lambda) \begin{pmatrix} C_{11} & 0 \\ 0 & C_{22} \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \tag{3.3.24}$$

Its eigenvalues come in pairs: $\{1-\lambda_1, 1+\lambda_1, ..., 1-\lambda_n, 1+\lambda_n\}$. For invertible B, the eigenvalues of a generalized eigenvector problem $Av = \lambda Bv$ are the same

as the eigenvalues of the eigenvector problem $B^{-1}Av = \lambda v$. It follows that the product of the eigenvalues can be expressed as the ratio of the determinants, i.e. $\prod_i \lambda_i = \det(B^{-1}A) = \frac{\det(A)}{\det(B)}$. So the ratio of determinants $\frac{\det C}{\det C_{11} \det C_{22}}$ is equal to the product of the generalized eigenvalues of Equation 3.3.24. Therefore we can rewrite the mutual information between the two multivariate Gaussian random variables X and Y from Proposition 3.3.6 as

$$I(X,Y) = -\frac{1}{2}\log\prod_{i=1}^{n}(1-\lambda_i)(1+\lambda_i) = -\frac{1}{2}\sum_{i=1}^{n}\log(1-\lambda_i^2) \tag{3.3.25}$$

This relationship between the mutual information and the eigenvalues of the covariance matrices gives the motivation to define a new dependence measure: the Kernel Generalized Variance. Instead of using only the largest eigenvalue of the generalized eigenvector problem 3.3.20 as in the $\mathcal{F}$-correlation case, we use all of the eigenvalues for the Kernel Generalized Variance.

Definition 3.3.20. (Kernel Generalised Variance): Let $X \in \mathcal{X} \subset \mathbb{R}^p$ and $Y \in \mathcal{Y} \subset \mathbb{R}^q$ be two random variables. Let $\mathcal{F}$ and $\mathcal{G}$ be the Hilbert spaces of functions from $\mathcal{X}$ and $\mathcal{Y}$ respectively to $\mathbb{R}$. Suppose we have a finite number of observations $(x,y) = ((x_1,y_1),...,(x_n,y_n))$ from X and Y. Let k and l be two reproducing kernels with corresponding feature maps $\phi : \mathcal{X} \to \mathcal{F}$ and $\psi : \mathcal{Y} \to \mathcal{G}$ and the kernel matrices K and L, defined as $K_{ij} = k(x_i, x_j)$ and $L_{ij} = l(y_i, y_j)$. Then the *Kernel Generalised Variance* between x and y is defined as

$$\mathrm{KGV}(x,y) = -\frac{1}{2}\log\frac{\det C}{\det D} = -\frac{1}{2}\sum_{i=1}^{n}\log(1-\lambda_i^2)$$

where $C = \begin{pmatrix} \left(\tilde{K} + \frac{p\kappa}{2}I\right)^2 & \tilde{K}\tilde{L} \\ \tilde{L}\tilde{K} & \left(\tilde{L} + \frac{q\kappa}{2}I\right)^2 \end{pmatrix}$, $D = \begin{pmatrix} \left(\tilde{K} + \frac{p\kappa}{2}I\right)^2 & 0 \\ 0 & \left(\tilde{L} + \frac{q\kappa}{2}I\right)^2 \end{pmatrix}$ and λ_i's are the generalized eigenvalues of $Cv = \lambda Dv$.

Note that in the above definition we provided the regularised version. If X and Y are multivariate Gaussian random variables the KGV is exactly their mutual information. In this section we have introduced two independence criteria, namely the Kernel Canonical Correlation and the Kernel Generalised Variance. We have provided the motivation to use these independence criteria as well as their formal definitions and empirical estimates.

3.4 Hilbert-Schmidt Independence Criterion

In this section we discuss the Hilbert-Schmidt Independence Criterion (HSIC) which was introduced in Gretton et al. (2005) and Gretton et al. (2008). This is the last independence criterion from the literature that we are introducing. We chose to present it last as it is the most theoretically involved out of the three independence criteria (dCov, $\mathcal{F}$-correlation and HSIC) but it provides more flexibility than the others. It relies on the same theoretical background as the canonical correlation - the Reproducing Kernel Hilbert Space. Necessary theoretical background is provided in Section 3.3.3. It also uses the theory of the Hilbert-Schmidt spaces, required definitions are provided in the following section.

We are not providing a motivational example for the HSIC as it follows the same ideas as for the Kernel Canonical Correlation in Section 3.3.2. Given two random variables $X \in \mathcal{X}$ and $Y \in \mathcal{Y}$, if we can find two functions $f : \mathcal{X} \to \mathbb{R}$ and $g : \mathcal{Y} \to \mathbb{R}$ such that $\text{Cov}(f(X), g(Y))$ is significantly different from zero, we conclude that random variables X and Y are not independent.

We start this section by giving the necessary background for the Hilbert-Schmidt spaces, then present the HSIC and its empirical estimate. Lastly we provide some of the asymptotic properties of the HSIC.

3.4.1 The Hilbert-Schmidt Space

Lets consider two Hilbert spaces $\mathcal{F}$ and $\mathcal{G}$ such that $\mathcal{F} = \{f \text{ s.t. } f : \mathcal{X} \to \mathbb{R}\}$ and $\mathcal{G} = \{g \text{ s.t. } g : \mathcal{Y} \to \mathbb{R}\}$. Then for each element $x \in \mathcal{X}$ there is a corresponding element $\phi_x(\cdot) \in \mathcal{F}$ such that $\langle \phi_x, \phi_{x'} \rangle_{\mathcal{F}} = k(x, x')$, where $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a unique positive definite kernel. We require that $\mathcal{F}$ is separable (it is sufficient that the kernel is continuous and $\mathcal{X}$ is separable). The second RKHS $\mathcal{G}$, with kernel $l(\cdot, \cdot)$ and feature map ψ on the separable space $\mathcal{Y}$ is defined similarly.

As the RKHS $\mathcal{F}$ and $\mathcal{G}$ are separable, they have countable orthonormal basis by the Lemma 3.3.1. Now we have the required notation to provide the definition for the Hilbert-Schmidt norm and operator.

Definition 3.4.1. (Hilbert-Schmidt norm): Let $C : \mathcal{G} \to \mathcal{F}$ be a linear operator. Then, provided the sum converges, *the Hilbert-Schmidt (HS) norm* of C is defined as

$$\|C\|_{HS}^2 := \sum_{i,j} \langle C v_i, u_j \rangle_{\mathcal{F}}^2 \tag{3.4.1}$$

where u_i and v_j are the orthonormal bases of $\mathcal{F}$ and $\mathcal{G}$ respectively.

Definition 3.4.2. (Hilbert-Schmidt operator): Let $C : \mathcal{G} \to \mathcal{F}$ be a linear operator. C is a *Hilbert-Schmidt operator* if its HS norm exists. The *Hilbert-Schmidt space* is a separable Hilbert space of the Hilbert-Schmidt operators $\mathrm{HS}(\mathcal{G}, \mathcal{F}) : \mathcal{G} \to \mathcal{F}$ with the inner product

$$\langle C, D \rangle_{HS} := \sum_{i,j} \langle Cv_i, u_j \rangle_{\mathcal{F}} \langle Dv_i, u_j \rangle_{\mathcal{F}} \tag{3.4.2}$$

The HSIC relies on a special Hilbert Schmidt operator - the Cross Covariance Operator. Before we can define it, we first need to introduce the *Tensor Product Operator*. Consider two functions $f \in \mathcal{F}$ and $g \in \mathcal{G}$. Then the tensor product operator $f \otimes g : \mathcal{G} \to \mathcal{F}$ is defined as

$$(f \otimes g)h := f\langle g, h \rangle_{\mathcal{G}} \text{ for all } h \in \mathcal{G} \tag{3.4.3}$$

Before we can define the cross covariance operator we first state and prove a result about the HS norm of the tensor products.

Proposition 3.4.1. (HS norm of Tensor Products): The Hilbert-Schmidt inner product of two tensor products can be rewritten in the following way

$$\langle f \otimes g, h \otimes k \rangle_{HS} = \langle f, h \rangle_{\mathcal{F}} \langle g, k \rangle_{\mathcal{G}} \tag{3.4.4}$$

Proof. Proof is provided in the Appendix C, Proposition C.1.5. □

Another important concept is the *mean element* of the Hilbert space $\mathcal{F}$. Let $(\mathcal{X}, \Gamma, p_X)$ and $(\mathcal{Y}, \Lambda, p_Y)$ be two probability spaces, i.e. $\mathcal{X}$ and $\mathcal{Y}$ are two separable spaces with the Borel sets Γ and Λ and probability measures p_X and p_Y respectively. Let $\mathcal{F}$ and $\mathcal{G}$ be two Hilbert spaces of functions from $\mathcal{X}$ and $\mathcal{Y}$ respectively to $\mathbb{R}$ (as defined before). Let $\phi : \mathcal{X} \to \mathcal{F}$ and $\psi : \mathcal{Y} \to \mathcal{G}$ be two feature maps corresponding to some reproducing kernels. Then we define the mean elements $\mu_x(\cdot) \in \mathcal{F}$ and $\mu_y(\cdot) \in \mathcal{G}$ with respect to these measures as

$$\begin{aligned} \langle \mu_X, f \rangle_{\mathcal{F}} &:= \mathrm{E}_X\Big[\langle \phi_X, f \rangle_{\mathcal{F}}\Big] = \mathrm{E}_X\Big[f(X)\Big] \quad \forall f \in \mathcal{F} \\ \langle \mu_Y, g \rangle_{\mathcal{G}} &:= \mathrm{E}_Y\Big[\langle \psi_Y, g \rangle_{\mathcal{G}}\Big] = \mathrm{E}_Y\Big[g(Y)\Big] \quad \forall g \in \mathcal{G} \end{aligned} \tag{3.4.5}$$

That is, μ_x is the element of $\mathcal{F}$ such that its inner product with any $f \in \mathcal{F}$ is equal to the expected value of f over the probability space $(\mathcal{X}, \Gamma, p_X)$.
It follows that

$$\begin{aligned}\|\mu_x\|_{\mathcal{F}}^2 = \langle \mu_X, \mu_X \rangle_{\mathcal{F}} &= \mathrm{E}_X\Big[\langle \phi_X, \mu_X \rangle_{\mathcal{F}}\Big] \\ &= \mathrm{E}_X\Big[\mathrm{E}_{X'}\Big[\langle \phi_X, \phi_{X'} \rangle_{\mathcal{F}}\Big]\Big] = \mathrm{E}_{X,X'}\Big[\langle \phi_X, \phi_{X'} \rangle_{\mathcal{F}}\Big] \\ &= \mathrm{E}_{X,X'}\Big[k(X, X')\Big]\end{aligned} \tag{3.4.6}$$

Here we treat the random variables X and X' as two independent copies of a random variable with the same distribution.
Now we have all the required pieces to define the essential object of the HSIC, namely the *Cross-Covariance Operator*.

Definition 3.4.3. (Cross-Covariance operator): The cross-covariance operator associated with the joint measure p_{xy} on $(\mathcal{X} \times \mathcal{Y}, \Gamma \times \Lambda)$ is a linear operator $C_{XY} : \mathcal{G} \to \mathcal{F}$ defined as

$$\langle f, C_{XY} g \rangle_{\mathcal{F}} = \mathrm{E}_{XY}\Big[\left(f(X) - \mathrm{E}_X[f(X)]\right) \left(g(Y) - \mathrm{E}_Y[g(Y)]\right) \Big] \tag{3.4.7}$$

Note that this is equivalent to $\langle f, C_{XY} g \rangle_{\mathcal{F}} = \mathrm{Cov}(f(X), g(Y))$, as it was defined in the previous section. Using the following explicit calculation

$$\begin{aligned}\langle f, C_{XY} g \rangle_{\mathcal{F}} &= \langle f, \mathrm{E}_{xy}\Big[(\phi_X - \mu_X) \otimes (\psi_Y - \mu_Y)\Big] g \rangle_{\mathcal{F}} \\ &= \mathrm{E}_{XY} \langle f, (\phi_X - \mu_X) \langle \psi_Y - \mu_Y, g \rangle_{\mathcal{G}} \rangle_{\mathcal{F}} \\ &= \mathrm{E}_{XY}\Big[\left(\langle f, \phi_X \rangle_{\mathcal{F}} - \langle f, \mu_X \rangle_{\mathcal{F}}\right) \left(\langle g, \psi_Y \rangle_{\mathcal{G}} - \langle g, \mu_Y \rangle_{\mathcal{G}}\right) \Big] \\ &= \mathrm{E}_{XY}\Big[\left(f(X) - \mathrm{E}_X[f(X)]\right) \left(g(Y) - \mathrm{E}_Y[g(Y)]\right) \Big]\end{aligned}$$

We can write the cross-covariance operator itself as

$$C_{XY} := \mathrm{E}_{XY}\Big[(\phi_X - \mu_X) \otimes (\psi_Y - \mu_Y)\Big] \tag{3.4.8}$$

3.4.2 Hilbert-Schmidt Independence Criterion

Now we introduce the Hilbert-Schmidt Independence Criterion as it was presented in Gretton et al. (2005) and Gretton et al. (2008).

Definition 3.4.4. (HSIC): Given separable RKHSs $\mathcal{F}$ and $\mathcal{G}$ and a joint probability measure p_{xy} over $(\mathcal{X} \times \mathcal{Y}, \Gamma \times \Lambda)$, we define the Hilbert-Schmidt Independence Criterion

(HSIC) as the squared HS-norm of the associated cross-covariance operator C_{xy}:

$$\mathrm{HSIC}(p_{xy}, \mathcal{F}, \mathcal{G}) := \|C_{xy}\|^2_{HS} \tag{3.4.9}$$

This definition is not very convenient for actually calculating the HSIC as it involves the tensor product and the mean elements of the Hilbert spaces which are not intuitive quantities. The following Lemma gives a convenient way of finding the HSIC in terms of kernels.

Lemma 3.4.2 (Gretton et al. (2005))**.** The Hilbert-Schmidt Independence Criterion can be expressed in terms of kernels in the following way:

$$\begin{aligned}\mathrm{HSIC}(p_{XY}, \mathcal{F}, \mathcal{G}) = \mathrm{E}_{X,Y,X',Y'}\Big[k(X,X')l(Y,Y')\Big] \\ - 2\mathrm{E}_{X,Y}\Big[\mathrm{E}_{X'}\Big[k(X,X')\Big]\mathrm{E}_{Y''}\Big[l(Y,Y'')\Big]\Big] \\ + \mathrm{E}_{X,X'}\Big[k(X,X')\Big]\mathrm{E}_{Y,Y'}\Big[l(Y,Y')\Big]\end{aligned} \tag{3.4.10}$$

Proof. The proof can be found in the original paper, we also provide a more explicit version of the proof in the Appendix C, Lemma C.1.6. □

3.4.3 Empirical Estimate of HSIC

So far we have introduced the necessary terminology of the Hilbert-Schmidt spaces and defined the Hilbert-Schmidt Independence Criterion. In this section we will provide an empirical estimate of the HSIC.

Definition 3.4.5. Let $(x, y) = ((x_1, y_1), ..., (x_n, y_n)) \subseteq (\mathcal{X} \times \mathcal{Y})^n$ be a series of n independent observations drawn from p_{XY}. An unbiased estimator of the $\mathrm{HSIC}(p_{XY}, \mathcal{F}, \mathcal{G})$, written $\mathrm{HSIC}(x, y)$, is given by a sum of three U-statistics

$$\mathrm{HSIC}(x,y) := \frac{1}{(n)_2}\sum_{(i,j)\in i_2^n} k_{ij}l_{ij} + -2\frac{1}{(n)_3}\sum_{(i,j,q)\in i_3^n} k_{ij}l_{iq} + \frac{1}{(n)_4}\sum_{(i,j,q,r)\in i_4^n} k_{ij}l_{qr} \tag{3.4.11}$$

Here $(n)_k = \frac{n!}{(n-k)!}$ and i_k^n are all k-tuples drawn from $1, 2, ..., n$ without replacement. A biased estimator of HSIC, written $\mathrm{HSIC}_b(x, y)$, is given by as a sum of three V-Statistics

$$\begin{aligned}\mathrm{HSIC}_b(x,y) &:= \frac{1}{n^2}\sum_{i,j=1}^{m} k_{ij}l_{ij} + -2\frac{1}{n^3}\sum_{i,j,q=1}^{m} k_{ij}l_{iq} + \frac{1}{n^4}\sum_{i,j,q,r=1}^{m} k_{ij}l_{qr} \\ &= \frac{1}{n^2}\operatorname{Tr} KHLH\end{aligned} \tag{3.4.12}$$

here $H, K, L \in \mathbb{R}^{n \times n}$, $K_{ij} = k_{ij} = k(x_i, x_j)$, $L_{ij} = l_{ij} = l(y_i, y_j)$ and $H_{ij} = \delta_{ij} - n^{-1}$.

For a detailed presentation of the U-statistics, see Appendix, Section C.2. The HSIC estimate in Equation 3.4.11 is unbiased as all three U-statistics are unbiased estimates: each summand comes from independent samples. The unbiased estimate of $\mathrm{E}_{X,Y}\Big[\mathrm{E}_{X'}\Big[k(X,X')\Big]\mathrm{E}_{Y''}\Big[l(Y,Y'')\Big]\Big]$ is $\frac{1}{(m)_3}\sum_{(i,j,q)\in i_3^m} k_{ij}l_{iq}$. We want (x, y), x' and y' to come from three different samples as the expectation over X, Y is over p_{XY} (therefore x and y come from the same sample i) and expectations over X' and Y' are over the independent marginal probability distributions p_X (from sample j) and p_Y (from sample q). Similarly we want two independent samples to estimate $\mathrm{E}_{X,Y,X',Y'}\Big[k(X,X')l(Y,Y')\Big]$ and four independent samples to estimate $\mathrm{E}_{X,X'}\Big[k(X,X')\Big]\mathrm{E}_{Y'',Y'''}\Big[l(Y'',Y''')\Big]$. The difference between the unbiased and biased estimates is that in the biased estimate we allow the indices of the different sums to be equal.

Theorem 3.4.3 (Gretton et al. (2005))**.** Let E_{XY} denote the expectation taken over n independent samples (x_i, y_i) drawn from p_{XY}. Then

$$\mathrm{HSIC}(p_{xy}, \mathcal{F}, \mathcal{G}) = \mathrm{E}_{XY}\Big[\mathrm{HSIC}_b(X,Y)\Big] + O(n^{-1}) \tag{3.4.13}$$

Proof. The proof can be found in the original paper, we also provide a more explicit version of a proof in the Appendix C, Theorem C.1.7. □

We have shown that $\frac{1}{m^2}\operatorname{Tr} KHLH$ is a good empirical estimate of the HSIC. Recall that the trace is the sum of the eigenvalues of the matrix. So the empirical estimate of the HSIC is the (normalized) sum of the eigenvalues of $KHLH$, we should note the similarities with the kernel canonical correlation which used the largest eigenvalue and kernel generalized variance which used the product of the eigenvalues of the matrix $HKHHLH$ respectively as their empirical estimate.

3.4.4 Asymptotic Results

In this subsection we provide an asymptotic approximation for the HSIC estimate, for a statistical test of independence from a finite sample.
First define an auxiliary variable h_{ijqr}:

$$h_{ijqr} = \frac{1}{4!}\sum_{(t,u,v,w)\in\mathcal{P}(i,j,q,r)} k_{tu}l_{tu} - 2k_{tu}l_{tv} + k_{tu}l_{vw} \tag{3.4.14}$$

Where $\mathcal{P}(i,j,q,r)$ denotes all the permutations of integers (i,j,q,r). Then we can write $\mathrm{HSIC}_b(x,y)$ as

$$\text{HSIC}_b(x,y) = \frac{1}{m^4}\sum_{i,j,q,r=1}^{n} h_{ijqr} \tag{3.4.15}$$

and the U-statistic $\text{HSIC}_s(x,y)$ corresponding to $\text{HSIC}_b(x,y)$ as

$$\text{HSIC}_s(x,y) = \frac{1}{(m)_4}\sum_{i,j,q,r=1}^{m} h_{ijqr} \tag{3.4.16}$$

Note that for a large m

$$\text{HSIC}_b(x,y) = \frac{m(m-1)(m-2)(m-3)}{m^4}\text{HSIC}_s(x,y) = \text{HSIC}_s(x,y) + O(m^{-1}) \tag{3.4.17}$$

Recall that we are working in the following set up. Suppose we have two random variables $X \in \mathcal{X} \subset \mathbb{R}^m$ and $Y \in \mathcal{Y} \subset \mathbb{R}^m$ and a finite set of observations $(x,y) = ((x_1,y_1),...,(x_n,y_n))$. We are interested in testing

$$\mathcal{H}_0 : f_{X,Y} = f_X f_Y \qquad \text{vs.} \qquad \mathcal{H}_1 : f_{X,Y} \neq f_X f_Y \tag{3.4.18}$$

i.e. the null hypothesis "X and Y are independent" v.s. the $\mathcal{H}_1$ "X and Y are not independent". The following two theorems (Theorems 1 and 2 from Gretton et al. (2008)) provide the asymptotic results for the $\text{HSIC}_b(X,Y)$ under both $\mathcal{H}_0$ and $\mathcal{H}_1$.

Theorem 3.4.4. Under the $\mathcal{H}_1$, $\text{HSIC}_b(X,Y)$ converges in distribution as $m \to \infty$ to a Gaussian according to

$$\sqrt{m}(\text{HSIC}_b(X,Y) - \text{HSIC}(p_{XY},\mathcal{F},\mathcal{G})) \to \mathcal{N}(0,\sigma_u^2) \tag{3.4.19}$$

where

$$\sigma_u^2 = 16\left(\text{E}_i\left[\text{E}_{jqr}[h_{ijqr}]\right]^2 - \text{HSIC}\left(p_{XY},\mathcal{F},\mathcal{G}\right)\right) \tag{3.4.20}$$

Proof. Special case of the Theorem C.2.3 in the Appendix, Section C.2. □

Theorem 3.4.5. Under the $\mathcal{H}_0$, the U-statistic $\text{HSIC}_s(x,y)$ is degenerate, that is $\text{E}_i[h_{ijqr}] = 0$. In this case, $\text{HSIC}_b(x,y)$ converges in distribution according

$$n\text{HSIC}_b(X,Y) \to \sum_{k=1}^{\infty}\lambda_k Z_k^2 \tag{3.4.21}$$

where $Z_k \sim \mathcal{N}(0,1)$ i.i.d., and λ_k are the solutions to the eigenvalue problem

$$\lambda_l \psi_l(z_i) = \mathrm{E}_{Z_j, Z_q, Z_r}\Big[h_{ijqr}(z_i, Z_j, Z_q, Z_r)\psi_l(z_i)\Big] \tag{3.4.22}$$

Proof. Special case of the Theorem C.2.4 in the Appendix, Section C.2. □

We should note that Székely et al. (2009) provide similar results to Theorems 3.4.4 & 3.4.5 for the distance covariance. This comes as no surprise, as we will see in the following section, there is a very close relationship between the distance covariance and the HSIC.
Under the null hypothesis, that is assuming that X and Y are independent, we can approximate the null distribution (the infinite sum of chi-squared random variables) by the two parameter Gamma distribution.

$$\mathrm{HSIC}_b(X,Y) \sim \Gamma(\alpha, \beta) \tag{3.4.23}$$

where

$$\alpha = \frac{(\mathrm{E}(\mathrm{HSIC}_b(X,Y))^2}{\mathrm{Var}(\mathrm{HSIC}_b(X,Y))} \text{ and } \beta = \frac{\mathrm{Var}(\mathrm{HSIC}_b(X,Y))}{\mathrm{E}(\mathrm{HSIC}_b(X,Y))} \tag{3.4.24}$$

Theorem 3.4.6 (Gretton et al. (2008))**.** (Mean of $\mathrm{HSIC}_b(X,Y)$): Under $\mathcal{H}_0$ the mean of $\mathrm{HSIC}_b(X,Y)$ is

$$\mathrm{E}(\mathrm{HSIC}_b(X,Y)) = m^{-1}\left(\mathrm{E}_{XY} kl - \mathrm{E}_X k \|\mu_Y\|^2 - \mathrm{E}_Y l \|\mu_X\|^2 + \|\mu_X\|^2 \|\mu_Y\|^2\right) \tag{3.4.25}$$

Here $\mathrm{E}_X k = \mathrm{E}_X[k(X,X)]$ and $\mathrm{E}_Y l = \mathrm{E}_Y[l(Y,Y)]$. $\|\mu_X\|^2 = \mathrm{E}_{XX'} k(X, X')$, where X and X' are independent and identically distributed.

Proof. Proof can be found in the original paper, we also provide a more explicit version of a proof in the Appendix C.1.3, Theorem C.1.8. □

Theorem 3.4.7 (Gretton et al. (2008))**.** (Variance of $\mathrm{HSIC}_b(X,Y)$): Under $\mathcal{H}_0$ the variance of $\mathrm{HSIC}_b(X,Y)$ is

$$\mathrm{Var}(\mathrm{HSIC}_b(X,Y)) = \frac{2(m-4)(m-5)}{(m)_4} \|C_{xx}\|^2_{HS} \|C_{yy}\|^2_{HS} \tag{3.4.26}$$

Here $\|C_{xx}\|^2_{HS} = \frac{1}{m^2} \operatorname{Tr} KHKH + O(m^{-1})$.

Proof. Proof can be found in the original paper, we also provide a more explicit version of a proof in the Appendix C.1.3, Theorem C.1.9. □

Distance Kernel: For the distance kernel $k(x_i, x_j) = \|x_i - x_j\|$ Equation 3.4.25 simplifies to

$$\mathrm{E}(\mathrm{HSIC}_b(X, Y)) \approx m^{-1} \widehat{\|\mu_x\|}^2 \widehat{\|\mu_y\|}^2 \tag{3.4.27}$$

Gaussian Kernel: For the Gaussian kernel $k(x_i, x_j) = \exp\left(\frac{-\|x_i - x_j\|^2}{\sigma^2}\right)$ Equation (3.4.25) simplifies to

$$\mathrm{E}(\mathrm{HSIC}_b(X, Y)) \approx m^{-1} \left(1 - \widehat{\|\mu_x\|}^2\right) \left(1 - \widehat{\|\mu_y\|}^2\right) \tag{3.4.28}$$

Here $\widehat{\|\mu_x\|^2} = \frac{1}{(m)_2} \sum_{i,j \in i_m^2} k(x_i, x_j)$ and $\widehat{\|\mu_y\|^2} = \frac{1}{(m)_2} \sum_{i,j \in i_m^2} l(y_i, y_j)$. These approximations hold because for the distance kernel $\mathrm{E}_X[k(X, X)] = 0$ and for the Gaussian kernel $\mathrm{E}_X[k(X, X)] = 1$.

3.5 Similarities between the Independence Criteria

So far we have introduced three independence criteria: the distance covariance, the kernel canonical correlation (and its alternative the kernel generalized variance) and the Hilbert Schmidt Independence Criterion. They all had different motivation and the theoretical interpretation of these criteria was different. Despite this, there are quite a few similarities among them. First of all they all rely on the eigenvalues of the product of the kernel matrices. In the case of the distance covariance we use a special type of kernel - the Euclidean distance. For the kernel canonical correlation Bach and Jordan (2002) suggested to use the Gaussian kernel and finally there is no restriction for the kernel in the HSIC, therefore it could be considered as a generalized version.
In this section we discuss the similarities between the three previously introduced independence criteria. Sejdinovic et al. (2013) established the equivalence between the distance-based (e.g. dCov) and RKHS-based (e.g. HSIC) statistics to measure dependence. We on the other hand show explicitly that dCov and KGV can be treated as the special cases of the HSIC with specific kernels. These are less general results than what Sejdinovic et al. (2013) showed but provides more intuition into the relationship between the independence criteria.

3.5.1 Relationship between dCov and HSIC

We start by presenting the relationship between the distance covariance and the HSIC. dCov and HSIC are one of the most popular independence criteria therefore this relationship is of great interest. We use some of the ideas presented in Schölkopf (2001). To begin with we note the similarities between the empirical estimates of the dCov (Definition 3.2.2) $\frac{1}{n^2}\sum_{i,j=1}^{n}\tilde{K}_{ij}\tilde{L}_{ij}$ and HSIC (Definition 3.4.5) $\frac{1}{n^2}\operatorname{Tr} KHLH$, where $\tilde{K} = HKH$.
First note that $HH = H$ as

$$\begin{aligned} HH &= (I_n - n^{-1}\mathbf{1}\mathbf{1}^T)(I_n - n^{-1}\mathbf{1}\mathbf{1}^T) \\ &= I_n - n^{-1}\mathbf{1}\mathbf{1}^T - n^{-1}\mathbf{1}\mathbf{1}^T + n^{-2}\mathbf{1}n\mathbf{1}^T \\ &= I_n - n^{-1}\mathbf{1}\mathbf{1}^T = H \end{aligned} \tag{3.5.1}$$

Secondly recall that $\sum_{i,j=1}^{n}\tilde{K}_{ij}\tilde{L}_{ij} = \operatorname{Tr}\tilde{K}\tilde{L}$ (shown in Proposition 3.2.1). It follows that

$$\operatorname{Tr}\tilde{K}\tilde{L} = \operatorname{Tr}(HKH)(HLH) = \operatorname{Tr}(KHH)(LHH) = \operatorname{Tr} KHLH \tag{3.5.2}$$

We see that both empirical estimates have the same form up to the definition of the matrices K and L. Before we can prove that dCov is a special case of HSIC in Proposition 3.5.5, we need to state and prove some auxiliary results.

Definition 3.5.1. A symmetric function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ for which for all $n \in \mathbb{N}$, for all $x_1, x_2, ..., x_n \in \mathcal{X}$ and for all $c_1, c_2, ...c_n \in \mathbb{R}$ we have

$$\sum_{i,j=1}^{n} c_i c_j k(x_i, x_j) \geq 0$$

is called a *positive definite (pd) kernel*.

Definition 3.5.2. A symmetric function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ for which for all $n \in \mathbb{N}$, for all $x_1, x_2, ..., x_n \in \mathcal{X}$ and for all $c_1, c_2, ...c_n \in \mathbb{R}$, s.t. $\sum_{i=1}^{n} c_i = 0$ we have

$$\sum_{i,j=1}^{n} c_i c_j k(x_i, x_j) \geq 0$$

is called a *conditionally positive definite (cpd) kernel*.

Proposition 3.5.1. (Distance measure is a (cpd) kernel): The negative Euclidean squared distance $d^2 : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ defined as $d^2(x, y) = -\|x - y\|^2$ for all $x, y \in \mathbb{R}$ is a cpd kernel.

Proof. For any $n \in \mathbb{N}$, for all $x_1, x_2, ..., x_n \in \mathcal{X}$ and for all $c_1, c_2, ...c_n \in \mathbb{R}$, s.t. $\sum_{i=1}^n c_i = 0$ we have

$$
\begin{aligned}
&- \sum_{i,j=1}^{n} c_i c_j \|x_i - x_j\|^2 = - \sum_{i,j=1}^{n} c_i c_j \left(\|x_i\|^2 - 2\langle x_i, x_j \rangle + \|x_j\|^2 \right) \\
&= - \underbrace{\left(\sum_{j=1}^{n} c_j \right)}_{=0} \left(\sum_{i=1}^{n} c_i \|x_i\|^2 \right) + 2 \sum_{i,j=1}^{n} c_i c_j \langle x_i, x_j \rangle - \underbrace{\left(\sum_{i=1}^{n} c_i \right)}_{=0} \left(\sum_{j=1}^{n} c_j \|x_j\|^2 \right) \\
&= 2 \sum_{i,j=1}^{n} c_i c_j \langle x_i, x_j \rangle = 2 \| \sum_{i=1}^{m} c_i x_i \|^2 \geq 0
\end{aligned}
$$

□

Proposition 3.5.2 (Berg et al. (1984))**.** If $k : \mathcal{X} \times \mathcal{X} \to [-\infty, 0]$ is a cpd kernel, then so is $-(-k)^\alpha$, $\alpha \in (0, 1)$.

Corollary 3.5.1. Negative Euclidean distance is a cpd kernel.

Proof. The negative Euclidean squared distance $d^2(x, y) = -\|x - y\|^2$ is a cpd by Proposition 3.5.1, then by Proposition 3.5.2 the negative Euclidean distance defined as $d(x, y) = -(-d^2(x, y)^{1/2} = -\|x - y\|$ is also a cpd. □

Proposition 3.5.3 (Schölkopf (2001))**.** Let K be a symmetric matrix, let $\mathbf{1} \in \mathbb{R}^n$ be the vector of all ones, I be a $n \times n$ identity matrix and some vector $c \in \mathbb{R}^n$ satisfies $\mathbf{1}^T c = 1$, then

$$\tilde{K} := \left(I - c^T \mathbf{1} \right) K \left(I - c^T \mathbf{1} \right)$$

is positive definite iff K is conditionally positive definite.

Proof. "⇒" Suppose $\tilde{K}$ is positive definite, i.e. for any $a \in \mathbb{R}^n$, we have

$$0 \leq a^T \tilde{K} a = a^T K a + a^T \mathbf{1} c^T K c \mathbf{1}^T a - a^T K c \mathbf{1}^T a - a^T \mathbf{1} c^T K a$$

If $\sum_{i=1}^n a_i = a^T \mathbf{1} = \mathbf{1}^T a = 0$, then the three last terms vanish. So we have that $a^T K a \geq 0$, i.e. K is conditionally positive definite.
"⇐" Suppose K is conditionally positive definite. The map $(I - c\mathbf{1}^T)$ has its range in the orthogonal complement of $\mathbf{1}$, as for any $a \in \mathbb{R}^n$,

$$\mathbf{1}^T (I - c\mathbf{1}^T) a = \mathbf{1}^T a - \underbrace{\mathbf{1}^T c}_{=1} \mathbf{1}^T a = 0$$

Moreover, being symmetric and satisfying $(I - c\mathbf{1}^T)^2 = I - Ic\mathbf{1}^T - c\mathbf{1}^T I + c\underbrace{\mathbf{1}^T c}_{=1}\mathbf{1}^T = (I - c\mathbf{1}^T)$, the map $(I - c\mathbf{1}^T)$ is a projection. Thus $\tilde{K}$ is the restriction of K to the orthogonal complement of $\mathbf{1}$, and by definition of conditional positive definiteness, that is precisely the space where K is positive definite. □

Proposition 3.5.4 (Schölkopf (2001))**.** Let k be a symmetric kernel, $x_1, ..., x_n \in \mathcal{X}$ and let $c_1, ..., c_n \in \mathbb{R}$, s.t. $\sum_{i=1}^n c_i = 1$. Then

$$\tilde{k}(x, x') := \frac{1}{2}\left(k(x, x') - \sum_{i=1}^n c_i k(x, x_i) - \sum_{i=1}^n c_i k(x_i, x') + \sum_{i,j=1}^n c_i c_j k(x_i, x_j)\right)$$

is positive definite if and only if k is conditionally positive definite.

Proof. Consider a set of points $x'_1, ..., x'_{n'} \in \mathcal{X}$, $n' \in \mathbb{N}$ and let K be the $(n+n') \times (n+n')$ Gram matrix based on $x_1, ..., x_n, x'_1, ..., x'_{n'}$. Apply proposition 3.5.3 using $c_{n+1} = ... = c_{n+n'} = 0$. □

Proposition 3.5.5. The empirical estimate of the distance covariance is a special case of the empirical estimate of the HSIC.

Proof. Let $x = (x_1, ..., x_m) \in \mathcal{X}^n$ and $y = (y_1, ..., y_m) \in \mathcal{X}^n$ be two samples, K and L be Euclidean distance matrices for x and y respectively, with elements $K_{ij} = \|x_i - x_j\|$ and $L_{ij} = \|y_i - y_j\|$, H be an $n \times n$ matrix, with elements $H_{ij} = \delta_{ij} - 1/n$. Define $\tilde{K} = HKH$, $\tilde{L} = HLH$. We define the empirical estimate for the distance covariance between x and y as $\mathrm{dCov}(x, y) = \frac{1}{n^2} \operatorname{Tr} \tilde{K}\tilde{L}$. $\tilde{K}_- = -HKH$, $\tilde{L}_- = -HLH$ are positive definite matrices by Propositions 3.5.1, 3.5.3 and 3.5.4. Using kernels $\tilde{K}_-$ and $\tilde{L}_-$ the Hilbert-Schmidt independence criterion for samples x and y can be defined as $\mathrm{HSIC}(\tilde{K}_-, \tilde{L}_-)$.

$$\begin{aligned}\mathrm{dCov}(x, y) &= \frac{1}{n^2} \operatorname{Tr} \tilde{K}\tilde{L} = \frac{1}{n^2} \operatorname{Tr}(HKH)(HLH) \\ &= \frac{1}{n^2} \operatorname{Tr}(H(-HKH)H)(H(-HLH)H) = \mathrm{HSIC}(\tilde{K}_-, \tilde{L}_-)\end{aligned}$$

□

So it follows that the dCov is the Hilbert-Schmidt independence criterion with a specific reproducing kernels represented by the Gram matrices $-HKH$ and $-HLH$ (K and L are matrices of Euclidean distances).

3.5.2 Relationship between KGV and HSIC

In the following section we point out some similarities between the Kernel Generalised Variance and the Hilbert-Schmidt Independence Criterion. Before we can do that we need to state and prove some results.

Proposition 3.5.6. (Alternative form of 3.3.20): Let $r_\kappa(K) = \left(\tilde{K} + \frac{n\kappa}{2} I\right)^{-1} \tilde{K} = \tilde{K}\left(\tilde{K} + \frac{n\kappa}{2} I\right)^{-1}$. Then the eigenvalue problem

$$\begin{pmatrix} I & r_\kappa(K) r_\kappa(L) \\ r_\kappa(L) r_\kappa(K) & I \end{pmatrix} \begin{pmatrix} \hat{\alpha} \\ \hat{\beta} \end{pmatrix} = (1+\lambda) \begin{pmatrix} \hat{\alpha} \\ \hat{\beta} \end{pmatrix} \tag{3.5.3}$$

is equivalent to the generalised eigenvalue problem 3.3.20, i.e. they both have the same eigenvalues.

Proof. First of all we will introduce some new notation $\tilde{K}_\kappa = \tilde{K} + \frac{n\kappa}{2} I$, $r_\kappa(K) = \tilde{K}_\kappa^{-1}\tilde{K} = \tilde{K}\tilde{K}_\kappa^{-1}$ and $\hat{\alpha} = \tilde{K}_\kappa \alpha$. Similarly for $\tilde{L}_\kappa$, $r_\kappa(L)$ and $\hat{\beta}$. Then

$$\begin{pmatrix} \tilde{K}_\kappa^2 & \tilde{K}\tilde{L} \\ \tilde{L}\tilde{K} & \tilde{L}_\kappa^2 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = (1+\lambda) \begin{pmatrix} \tilde{K}_\kappa^2 & 0 \\ 0 & \tilde{L}_\kappa^2 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$$

$$\begin{pmatrix} \tilde{K}_\kappa^{-1} & 0 \\ 0 & \tilde{L}_\kappa^{-1} \end{pmatrix} \begin{pmatrix} \tilde{K}_\kappa^2 & \tilde{K}\tilde{L} \\ \tilde{L}\tilde{K} & \tilde{L}_\kappa^2 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = (1+\lambda) \begin{pmatrix} \tilde{K}_\kappa^{-1} & 0 \\ 0 & \tilde{L}_\kappa^{-1} \end{pmatrix} \begin{pmatrix} \tilde{K}_\kappa^2 & 0 \\ 0 & \tilde{L}_\kappa^2 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$$

$$\begin{pmatrix} \tilde{K}_\kappa & \tilde{K}_\kappa^{-1}\tilde{K}\tilde{L} \\ \tilde{L}_\kappa^{-1}\tilde{L}\tilde{K} & \tilde{L}_\kappa \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = (1+\lambda) \begin{pmatrix} \tilde{K}_\kappa & 0 \\ 0 & \tilde{L}_\kappa \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$$

$$\begin{pmatrix} I & \tilde{K}_\kappa^{-1}\tilde{K}\tilde{L}\tilde{L}_\kappa^{-1} \\ \tilde{L}_\kappa^{-1}\tilde{L}\tilde{K}\tilde{K}_\kappa^{-1} & I \end{pmatrix} \begin{pmatrix} \tilde{K}_\kappa \alpha \\ \tilde{L}_\kappa \beta \end{pmatrix} = (1+\lambda) \begin{pmatrix} \tilde{K}_\kappa \alpha \\ \tilde{L}_\kappa \beta \end{pmatrix}$$

$$\begin{pmatrix} I & r_\kappa(K) r_\kappa(L) \\ r_\kappa(L) r_\kappa(K) & I \end{pmatrix} \begin{pmatrix} \hat{\alpha} \\ \hat{\beta} \end{pmatrix} = (1+\lambda) \begin{pmatrix} \hat{\alpha} \\ \hat{\beta} \end{pmatrix}$$

□

Note that this gives us an alternative form for the Kernel Generalised Variance as well. Namely:

$$\mathrm{KGV}(x,y) = -\frac{1}{2} \log \det \begin{pmatrix} I & r_\kappa(K) r_\kappa(L) \\ r_\kappa(L) r_\kappa(K) & I \end{pmatrix} \tag{3.5.4}$$

Proposition 3.5.7. Let $M = \begin{pmatrix} A & B \\ C & D \end{pmatrix}$ be a block matrix. Then its determinant can be written as

$$\det M = \det\left(AD - BD^{-1}CD\right)$$

Proof.

$$\begin{aligned}\det\begin{pmatrix} A & B \\ C & D \end{pmatrix} &= \det\begin{pmatrix} A & B \\ C & D \end{pmatrix}\det\begin{pmatrix} I & 0 \\ -D^{-1}C & I \end{pmatrix} \\ &= \det\left(A - BD^{-1}C\right)\det(D) = \det\left(AD - BD^{-1}CD\right)\end{aligned}$$

□

Definition 3.5.3. Let M be an $n \times n$ dimensional matrix, with elements m_{ij}, then

$$\|M\|_{HS} = \left(\sum_{i,j=1}^{n} m_{ij}^2\right)^{1/2}$$

is called its Hilbert-Schmidt or Frobenius norm.

Proposition 3.5.8. Let M be an $n \times n$ dimensional matrix, with elements m_{ij} and eigenvalues $\lambda_1, ..., \lambda_n$, then its Hilbert-Schmidt norm squared is equal to the sum of its eigenvalues squared, that is

$$\|M\|_{HS}^2 = \sum_{i=1}^{n} \lambda_i^2$$

Proof. If λ is an eigenvalue of M, i.e. $Mu = \lambda u$, for some eigenvector u, then λ^2 is the eigenvalue of the matrix MM^T, as $MM^T u = M\lambda u = \lambda^2 u$. Then, recalling that the trace of a matrix is equal to the sum of its eigenvalues, it follows that

$$\|M\|_{HS}^2 = \sum_{i,j=1}^{n} m_{ij}^2 = \mathrm{Tr}\left(MM^T\right) = \sum_{i=1}^{n} \lambda_i^2$$

□

Proposition 3.5.9. Let M be an $n \times n$ dimensional matrix with a small Hilbert-Schmidt norm, i.e. its eigenvalues are small, then

$$-\frac{1}{2}\log\det\left(I - MM^T\right) \approx \frac{1}{2}\mathrm{Tr}\left(MM^T\right) \tag{3.5.5}$$

Proof. If λ is an eigenvalue of M, i.e. $Mu = \lambda u$, for some eigenvector u, then λ^2 and $1 - \lambda^2$ are the eigenvalues of the matrices MM^T and $I - MM^T$ respectively.

$$\left(I - MM^T\right) u = u - \lambda^2 u = \left(1 - \lambda^2\right) u$$

If $|\epsilon| \ll 1$, then it follows from the Taylor expansion that

$$\log(1 - \epsilon) = \epsilon + o(\epsilon) \approx \epsilon$$

Now combining the above we have

$$-\frac{1}{2} \log \det \left(I - MM^T\right) = -\frac{1}{2} \sum_i \log \left(1 - \lambda_i^2\right) \approx \frac{1}{2} \sum_i \lambda_i^2 = \frac{1}{2} \operatorname{Tr} \left(MM^T\right)$$

□

Close to the independence, we have

$$\hat{\rho}_{\mathcal{F}} = \max_{f,g \in \mathcal{F}} \widehat{\operatorname{Cor}}(f(x)g(y)) \ll 1 \Rightarrow \lambda_{max} \ll 1 \tag{3.5.6}$$

So the conditions for Proposition 3.5.9 holds and we have:

$$\begin{aligned}
\operatorname{KGV}(x, y) &= -\frac{1}{2} \log \det \begin{pmatrix} I & r_\kappa(K) r_\kappa(L) \\ r_\kappa(L) r_\kappa(K) & I \end{pmatrix} \\
&= -\frac{1}{2} \log \det \left(I - r_\kappa(K) r_\kappa(L) r_\kappa(L) r_\kappa(K)\right) && \text{by Proposition 3.5.7} \\
&\approx \frac{1}{2} \operatorname{Tr} \left(r_\kappa(K)^2 r_\kappa(L)^2\right) && \text{by Proposition 3.5.9.} \\
&= \frac{1}{n} \operatorname{Tr} \left(\sqrt{\frac{n}{2}} r_\kappa(K)^2 \sqrt{\frac{n}{2}} r_\kappa(L)^2\right) \\
&= \operatorname{HSIC}\left(\sqrt{\frac{n}{2}} r_\kappa(K)^2, \sqrt{\frac{n}{2}} r_\kappa(L)^2\right)
\end{aligned}$$

So it follows that the KGV is approximately the Hilbert-Schmidt independence criterion with a specific reproducing kernels represented by the Gram matrices $\sqrt{\frac{n}{2}} r_\kappa(K)^2$ and $\sqrt{\frac{n}{2}} r_\kappa(L)^2$ instead of $\tilde{K}$ and $\tilde{L}$.

3.6 Statistical Tests of Independence

3.6.1 Introduction

We introduced three different approaches to estimate the general independence between two random variables. We also introduced their empirical estimates, i.e. the way to estimate how dependent a pair of random variables (X, Y) is, given a finite sample $(x, y) = ((x_1, y_1), ..., (x_n, y_n))$ of the random variables. All the empirical estimates give us a single number. We know that a value of zero would mean independence, but this can happen only if all our observations lie on an exact grid, which happens with zero probability. What we are actually interested in is testing the independence hypothesis, i.e. testing

$$\mathcal{H}_0 : X \perp Y \qquad \text{vs.} \qquad \mathcal{H}_1 : X \not\perp Y \tag{3.6.1}$$

As in the case of any hypothesis testing we are interested in the confidence with which we could reject H_0. That is, we are interested in a p-value, the probability to observe data as dependent as (x, y) if H_0 is true, i.e. if the random variables X and Y are truly independent.
We are interested in two tests, the unconditional independence test which test hypothesis as in Equation 3.6.1 and the conditional independence test that tests whether X and Y are independent given random variable(s) Z, i.e.

$$\mathcal{H}_0 : X \perp Y \mid Z \qquad \text{vs.} \qquad \mathcal{H}_1 : X \not\perp Y \mid Z \tag{3.6.2}$$

In this section we present statistical tests for both unconditional and conditional dependence, as well as compare the independence tests themselves. We consider the distance covariance and the HSIC only, as the $\mathcal{F}$-correlation and similarly the KGV are computationally very expensive. As we have shown before, the distance covariance is equivalent to the HSIC using a special kernel based on the Euclidean distance. Nevertheless we will keep referring to it as the distance covariance and we will refer to the HSIC with a Gaussian kernel as the HSIC.

3.6.2 Test of the Unconditional Independence

In this section we discuss the unconditional independence tests. We present two different approaches, namely the permutation test and the gamma test. Recall that we have a finite sample $(x, y) = ((x_1, y_1), (x_2, y_2), ..., (x_n, y_n))$ from a pair of random

variables X and Y. We are interested in testing $\mathcal{H}_0$: X and Y are independent versus $\mathcal{H}_1$: X and Y are not independent.

Permutation Test

The first test is a simple permutation test. The idea behind it is that permuting y removes any dependency between x and y. Therefore we can compare the $\mathrm{IC}(x, y)$ (here IC stands for our chosen independence criterion) with $\mathrm{IC}(x, y_\rho)$, where y_ρ is the permutation ρ applied to the sample y. We choose the number of permutations r, and create r permuted samples $y_{(j)} = \{y_{\rho_j(i)}\}$, $j = 1, \ldots, r$. Then the p-value p is the fraction of times the $\mathrm{IC}(x, y)$ is smaller than the $\mathrm{IC}(x, y_{(j)})$. If X and Y are truly independent, permuting y will not change much and we will get a large p-value p and therefore we will not be able to reject the null hypothesis (X and Y are independent). On the other hand if X and Y were not independent, $\mathrm{IC}(x, y)$ will be much larger than any of the $\mathrm{IC}(x, y_{(j)})$, we will get a very small p-value (i.e. probability to observe data as dependent as (x, y) given that X and Y are independent is very low) and we will be able to reject the H_0. Algorithm of the permutation independence test is provided in Algorithm 2.

Data: x, y, r
Result: p-value
Calculate $\mathrm{IC}_{r+1} = \mathrm{IC}(x, y)$;
for i *in* $1 : r$ **do**
  Permute y to get $y_{(i)}$;
  Calculate $\mathrm{IC}_i = \mathrm{IC}(x, y_{(i)})$
end
Calculate p-value as $p = \frac{\sum_{i=1}^{n} \mathbf{1}(\mathrm{IC}_{r+1} < \mathrm{IC}_i)}{r+1}$;
return p

Algorithm 2: Permutation test

Gamma Test

The second approach is to use the Gamma approximation of the HSIC under the null hypothesis as discussed in Section 3.4.4. The same approximation works for the dCov. The value of the asymptotic distribution of the empirical estimate $\mathrm{IC}(x, y)$ under the null hypothesis of independence is approximated by a Gamma distribution: $\mathrm{IC}(x, y) \sim$

$\text{Gam}(\alpha, \theta)$ where α is the shape parameter and θ is the scale parameter calculated as

$$\alpha = \frac{\text{E}[\hat{\text{IC}}_{X,Y}]^2}{\text{Var}(\hat{\text{IC}}_{X,Y})}, \qquad \theta = \frac{\text{Var}(\hat{\text{IC}}_{X,Y})}{\text{E}[\hat{\text{IC}}_{X,Y}]}$$

The p-value is then obtained as an upper-tail quantile of $\text{IC}(x, y)$.

Efficient calculation

Computing $\text{IC}(x, y)$ for both the dCov and HSIC is expensive with complexity $O(n^3)$ (we need to multiply $n \times n$ kernel matrices), n being the sample size. Naively calculating the empirical estimate $r + 1$ times for the permutation test is extremely slow. (Bach and Jordan, 2002) suggested an incomplete Cholesky factorization for finding the $\mathcal{F}$-correlation efficiently. Using the m steps factorization reduces the complexity to $O(nm^3)$ for a chosen $m < n$. In more detail, K and L are approximated by $\tilde{K} \approx U_x D_x U_x^T$, with the matrix of eigenvectors U_x and the matrix of eigenvalues D_x of size $n \times m$ and $m \times m$, respectively. Similarly $\tilde{L} \approx U_y D_y U_y^T$. This factorization is suitable since, with slowly decaying kernel functions, Gram matrices often have a low effective rank. Instead of permuting the entries of y and recalculating the Gram matrix, we exploit the one-to-one relationship between samples y_i and the rows and columns of the Gram matrix L, since $L(i, j) = l(y_i, y_j)$ (with a symmetric kernel function l). In the calculation of $H(x, y_{(i)})$ we use $L_{(j)} = P_j L P_j^T \approx P_j U_y D_y U_y^T P_j^T$, with a permutation matrix P_j permuting rows according to permutation ρ_j. This means that an incomplete Cholesky decomposition needs to be performed only once for all permuted datasets $y_{(i)}$, and consequent values $H(x, y_{(i)})$ can be obtained simply by permuting coordinates of the eigenvectors in U_y ($\tilde{K}$ is kept the same).

3.6.3 Test of the Conditional Independence

The unconditional independence test provides us with a tool to check for an independence between two random variables X and Y. But we also may be interested in checking for a conditional independence, for example within the PC algorithm (to be discussed in more detail in the next chapter) or if we are interested in a mutual rather than pairwise independence. That is, we want to test the hypothesis as in Equation 3.6.2.

We explore two approaches. The first, *permutation-cluster test* suggested in Gretton et al. (2009) and Tillman (2009), is based on a conditional version of the HSIC from the Fukumizu et al. (2008). The alternative test we propose here is based on the *residuals*.

It is simpler in that it only requires an unconditional version of the HSIC and can be readily applied to other independence criteria for which there is no conditional version readily available that allows integration of a conditioning set of variables, as in the case of the dCov.

Permutation-Cluster Test

Fukumizu et al. (2008) provide an estimator for a *conditional* version $\mathrm{HSIC}(X, Y \mid Z)$ of the HSIC for a sample set (x_i, y_i, z_i), $i = 1, \ldots, n$ as

$$H(x, y \mid z) = \frac{1}{2} \operatorname{Tr}(\tilde{K}\tilde{L} - 2\tilde{K}\tilde{M}\tilde{M}_\epsilon^{-2}\tilde{M}\tilde{L} + \tilde{K}\tilde{M}\tilde{M}_\epsilon^{-2}\tilde{M}\tilde{L}\tilde{M}\tilde{M}_\epsilon^{-2}\tilde{M}) \tag{3.6.3}$$

where $\tilde{K}$, and $\tilde{L}$ are defined as before for x, and y, $\tilde{M}$ is the analogous Gram matrix for z and $\tilde{M}_\epsilon = (\tilde{M} + \epsilon I_n)$. ϵ is a regularization parameter that needs to be carefully selected (see the Results section). The calculation of $\mathrm{HSIC}(X, Y \mid Z)$ is computationally very expensive, but the same simplifications can be used as in the unconditional case.
In order to obtain a p-value for rejecting independence based on the estimator (3.6.3) for the conditional HSIC criterion, the samples are clustered according to the Euclidean distance between the z coordinates of samples. Sample labels of y are only permuted within each cluster, thus ensuring that the permuted samples break dependency between x and y for an approximately fixed z but retain their dependency on z. For the clustering we use a k-means algorithm (Hartigan and Wong (1979) implemented in `R` function `kmeans()`). A larger number of clusters is desirable to achieve an almost constant z within each cluster. On the other hand, enough samples are required in each cluster to achieve a permutation of labels that breaks any conditional dependency between x and y. For the sample sizes considered here, good results were obtained with a constant cluster number of 10.

Residuals Test

As a simpler alternative to obtain p-values for the conditional HSIC we propose to test residuals for independence based on any unconditional test of independence. The residuals r_x and r_y are obtained by regressing x and y on z in a nonlinear fashion. The regression removes the dependencies between x and y due to z and consequently the residuals should be independent if $\mathbf{X} \perp_P \mathbf{Y} \mid \mathbf{Z}$. For regression we use a generalized additive model (GAM, see Hastie and Tibshirani (1986)) as implemented in the `R` function `gam()` in the library `mgcv` (Wood, 2004, 2011) with default settings). That is, we regress y on a set of variables x_i, $i \in \{1, \ldots, p\}$ as $y = f_0 + \sum_{i=1}^{p} f_i(x_i) + \varepsilon$ where f_i

are spline functions (selected by cross-validation) and ε is Gaussian noise. We have now the option of using either the permutation or Gamma test from Section 3.6.2 on the residuals.

3.6.4 Simulation Results for the Independence Tests

In this section we compare the power of various independence tests to find dependence in simulated data. We will also discuss the choice of parameters for these tests.
We expect the effectiveness of the independence criteria to depend crucially on the signal to noise ratio. We therefore tested the HSIC and dCov on 300 samples simulated from $Y \sim \sin(X) + \mathcal{N}(0, \sigma^2)$, $X \sim \mathcal{U}(0, 10)$ for varying noise levels σ^2 and signal range of 2 from -1 to 1. X and Y are dependent, hence independence should be rejected with low p-values. The simulated data are shown in Figure 3.6. Table 3.1 lists the p-values for combinations of methods from Section 3.6.2 and varying noise levels σ. All the p-values are a mean of 100 replications of the test. The size of the simulated sample is 300, as this is a reasonably typical sample size for the high-throughput experiments. At $\sigma = 10$ variables X and Y are effectively independent. As expected the p-value is less and less reliable for detecting dependency for samples with increasing noise levels. In this simple test both criteria, HSIC and dCov, behave similarly.

Test	$\sigma = 1$	$\sigma = 2$	$\sigma = 5$	$\sigma = 10$
HSIC permutation	< 0.001	0.04	0.37	0.48
HSIC gamma	4e-13	0.04	0.38	0.49
dCov permutation	< 0.001	0.06	0.37	0.47
dCov gamma	5e-08	0.07	0.37	0.54

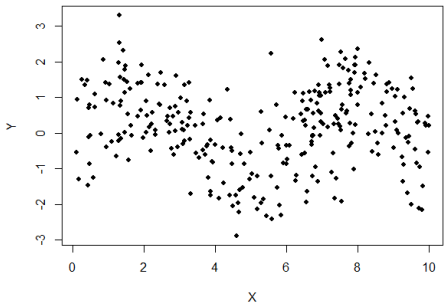
Table 3.1 Testing independence criteria. All the p-value estimates are the mean of 100 p-values from repetitions for each of the four tests. The size of the simulated sample is 300.

Both HSIC and dCov uses kernel matrices. The HSIC uses a Gaussian kernel which depends on the width parametes λ, i.e.

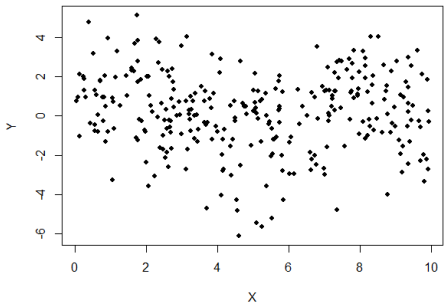
$$K(i, j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\lambda^2}\right) \tag{3.6.4}$$

and the dCov uses the distance kernel, which depends on the power parameter ξ, i.e.

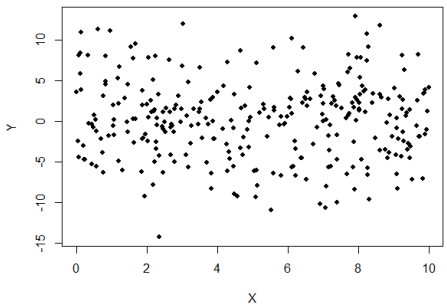
$$K(i, j) = \|x_i - x_j\|^{\xi} \tag{3.6.5}$$



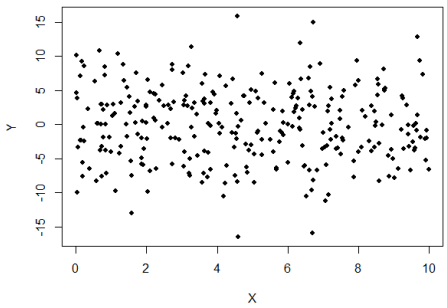
(a) $y = \sin(x) + \mathcal{N}(0,1)$, $x = \mathcal{U}(0,10)$.



(b) $y = \sin(x) + \mathcal{N}(0,4)$, $x = \mathcal{U}(0,10)$.



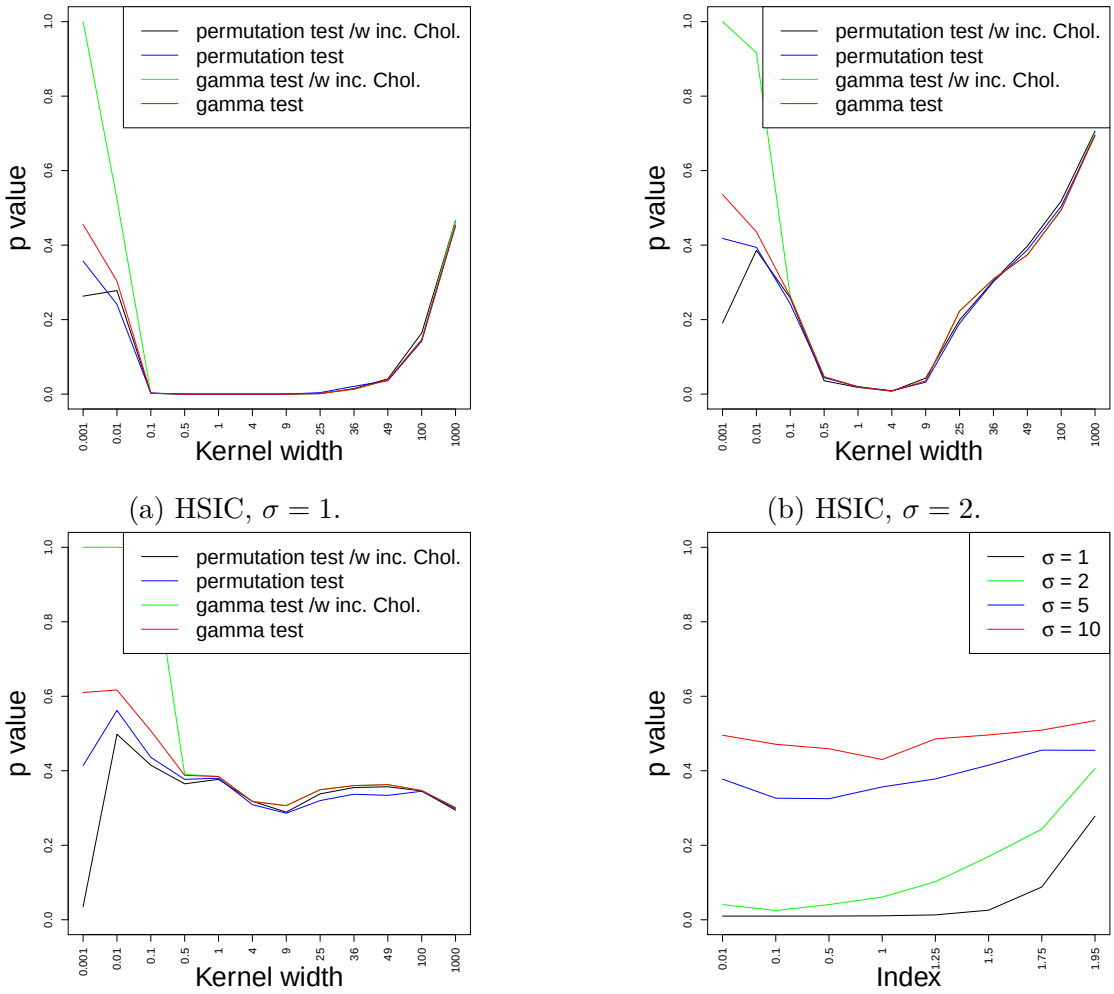
(c) $y = \sin(x) + \mathcal{N}(0,25)$, $x = \mathcal{U}(0,10)$.



(d) $y = \mathcal{N}(0,25)$, $x = \mathcal{U}(0,10)$.

Fig. 3.6 Data simulated with nonlinear dependencies

Figures 3.7a) to c) show p-values for different HSIC tests when the kernel width λ varies from 0.001 to 1000. We note that in order to reject independence successfully λ needs to be chosen carefully, particularly with higher noise. If λ is very small, then $k(x,y) \approx 0$ for almost all $x \neq y$, and the Gram matrix is close to the identity matrix. Gram matrix being close to the identity matrix also means that its eigenvalues do not decay fast enough and the incomplete Cholesky decomposition is not accurate, therefore the results can be misleading: p-values for different methods differ greatly. If λ is too large, then $k(x,y) \approx 1$, for all x, y and the Gram matrix is ill-conditioned. In this case any dependency between the variables is hard to detect. Based on these figures we choose a kernel width in the range $\lambda \in (0.5, 9)$. Note that HSIC is not scale invariant and so the suggested λ range has to be scaled appropriately in case we would choose to use a non-normalized data. In case we would choose to use a not normalized data, λ should be scaled appropriately. Furthermore we observe that the permutation and Gamma tests both with and without incomplete Cholesky decomposition always give very similar results for this range of λ. Therefore, for further analysis we will use



(a) HSIC, $\sigma = 1$.

(b) HSIC, $\sigma = 2$.

(c) HSIC, $\sigma = 10$.

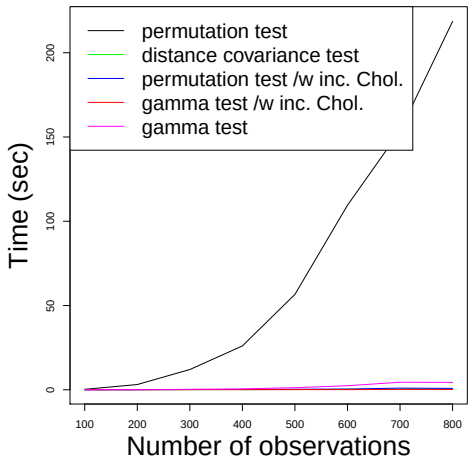
(d) dCov, $\sigma = 1, 2, 5, 10$.

Fig. 3.7 Dependency of p-values of HSIC and dCov tests on varying parameters, kernel width λ for HSIC and index ξ for dCov.

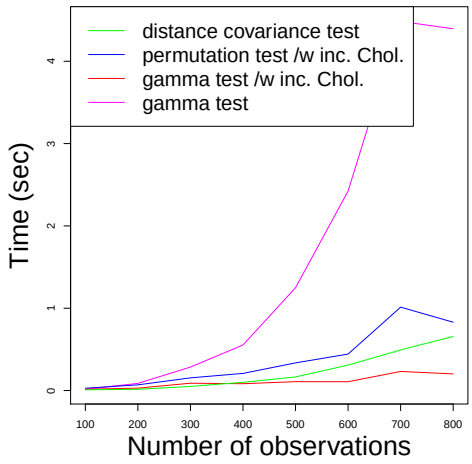
the Gamma test with an incomplete Cholesky decomposition since it is computationally most efficient (as seen in Figure 3.8).

Figure 3.7d) shows the dependency of p-values on the power parameter ξ of the dCov. For simplicity we set $\xi = 1$, although smaller values might work even better.

Finally Figure 3.9 show a large simulation of HSIC Gamma test only, number of observations changes from 100 to 1500, noise ϵ changes from 1 to 10 and the width parameter λ changes from 0.001 to 1000. The ability to detect dependence goes up with the number of observations and goes down with the increase in noise.



(a) Time taken by different methods to find the p-value



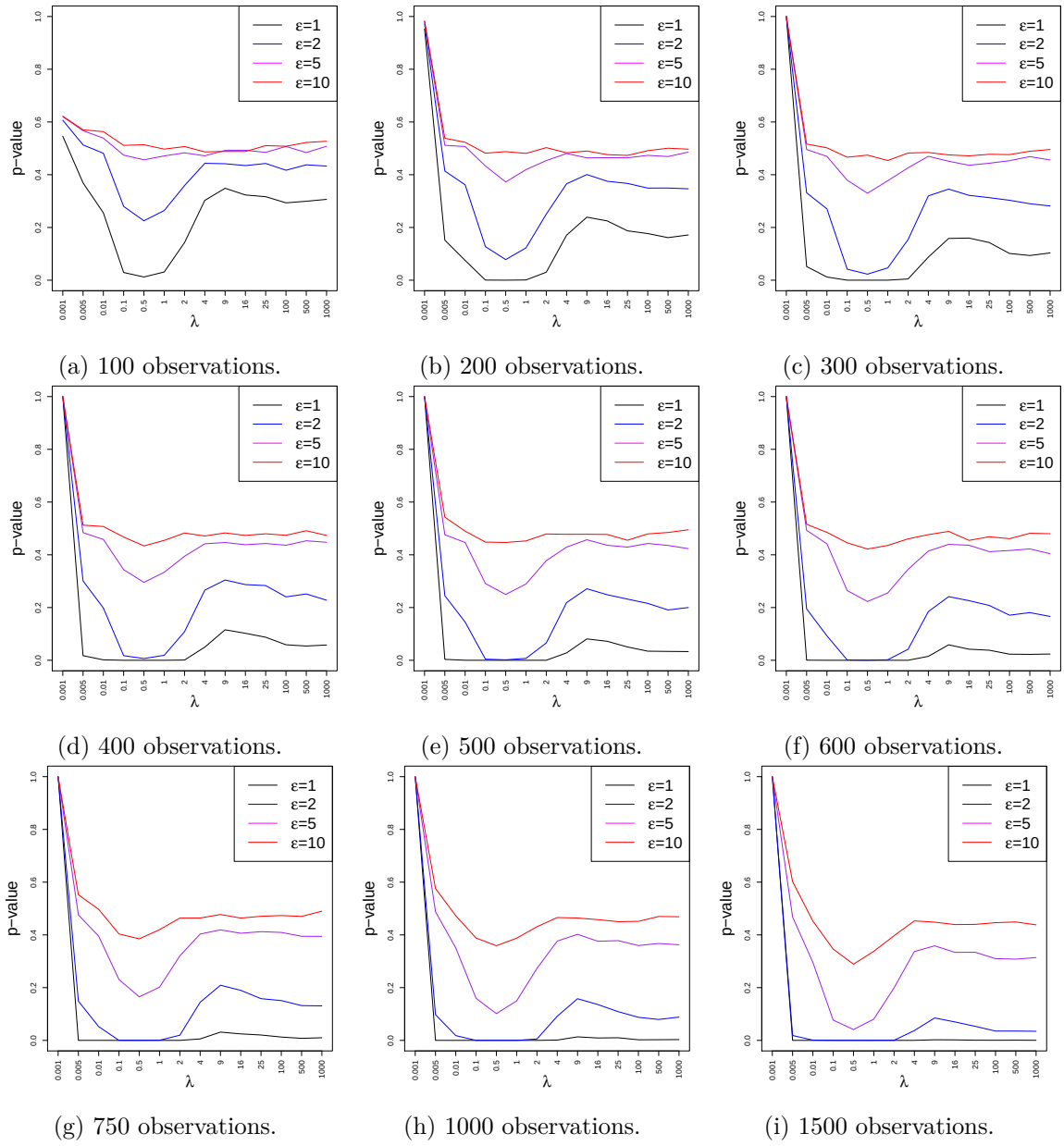
(b) Close up of Figure 3.8a without the HSIC permutation test.

Fig. 3.8 Time efficiency of independence tests.

3.7 Signal to Noise Ratio Independence Criterion

3.7.1 Introduction

We have introduced three independence criteria from the literature. In the essence they all rely on finding eigenvalues of some matrix. Given n observations we are dealing with $n^2 \times n^2$ kernel matrices. Multiplying such matrices or finding their trace or determinant might be time consuming (even when we use the incomplete Cholesky decomposition). Another consideration is that in our network inference models we are mainly considering the additive noise models which significantly restrict the form of the dependencies that two variables may have. Lastly in Section 3.6 we observed that the ability of the statistical independence tests highly depend on the ratio of the signal in the data to the noise. First two considerations raises a question: maybe we do not need a general independence criterion, maybe something simpler (and much faster to calculate) would work just as well? The last observation suggests that the answer might be in the signal to noise ratio. So we came up with a new independence criterion, namely the *Signal to Noise Ratio Independence Criterion* (or SNRIC).
In this section we will introduce the idea of the Signal to Noise Ratio IC and provide a theoretical and empirical motivation, why it might be an interesting quantity to consider in the light of the network inference. Finally we will provide some theoretical results to justify using it.



(a) 100 observations. (b) 200 observations. (c) 300 observations.

(d) 400 observations. (e) 500 observations. (f) 600 observations.

(g) 750 observations. (h) 1000 observations. (i) 1500 observations.

Fig. 3.9 Dependency of the p-values of the HSIC Gamma test on a varying kernel width parameter λ, noise level ϵ and the number of observations. Model: $Y \sim \sin(X) + \mathcal{N}(0, \sigma^2)$, $X \sim \mathcal{U}(0, 10)$.

3.7.2 Motivation

Theoretical motivation

Independence criterion, such as HSIC or dCov, is a great way of estimating relationship between data. However they do have their limitations. It tests the dependence between two variables X and Y given only a finite sample $(x, y) = ((x_1, y_1), ..., (x_n, y_n))$. This

implies that an empirical independence criteria will never (with probability 0) be exactly equal to zero, i.e. there will always be some dependence due to the finiteness of the sample. This suggests to explore the connection between the IC ability to detect dependence and the Signal to Noise Ratio (SNR) in the data. If the real SNR is too small we cannot expect the IC to detect dependence.
We assume a model $Y \sim f(X) + \epsilon$, where f is the signal and ϵ is the noise. After regressing Y on X we get $y_i = s_i + \epsilon_i$, where we assume $s_i = f(x_i)$ to be the signal and ϵ_i to be the noise. We use notation $S = \{s_1, ..., s_n\}$ and $\epsilon = \{\epsilon_1, ..., \epsilon_n\}$.
We consider two different signal to noise ratios, defined as:

- Variance based: $SNR_1 = \sqrt{\frac{\text{Var}(S)}{\text{Var}(\epsilon)}}$
- Amplitude based: $SNR_2 = \frac{\max(S)-\min(S)}{\max(\epsilon)-\min(\epsilon)}$

Linear relationship

First we considered the simplest relationship between X and Y - a linear relationship.

$$\begin{aligned} X &\sim U[0,1], \\ Y &\sim SX + U[0,1]. \end{aligned} \tag{3.7.1}$$

In our simulations the signal S varies from 1 to 2^{-5}, i.e. $S \in \{1, 2^{-1}, ..., 2^{-5}\}$. Sample size is 900.
The empirical estimate of SNR_1 given signal S is:

$$\begin{aligned} \text{Var}(\epsilon) &\approx \text{Var}(U[0,1]) = 1/12 \\ \text{Var}(S) &\approx \text{Var}(SX) \approx \text{Var}(U[0,S]) = S^2/12 \\ \text{SNR}_1(S) &\approx S \end{aligned} \tag{3.7.2}$$

The empirical estimate of SNR_2 given signal S is:

$$\begin{aligned} \max(\epsilon) - \min(\epsilon) &\approx \max(U[0,1]) - \min(U[0,1]) = 1 \\ \max(S) - \min(S) &\approx \max(U[0,S]) - \min(U[0,S]) = S \\ \text{SNR}_2(S) &\approx S \end{aligned} \tag{3.7.3}$$

We observe the expected results, shown in Table 3.2. If the SNR is smaller than some threshold (in this case around 0.1) we do not observe dependence and the statistical tests cannot reject the null hypothesis (variables are independent) any more.

S	dCov perm	dCov gamma	HSIC perm	HSIC gamma	SNR_1	SNR_2
1	0.0099	0	0	0	1.006	0.977
1/2	0.0099	0	0	0	0.505	0.490
1/4	0.0099	0	0	0	0.265	0.247
1/8	0.0219	0.0108	0.0154	0.0151	0.134	0.127
1/16	0.1907	0.2039	0.1943	0.1943	0.077	0.070
1/32	0.3827	0.3958	0.3854	0.3766	0.055	0.050

Table 3.2 Relationship between independence criteria and the SNR's for the linear relationship with varying noise levels.

S	dCov perm	dCov gamma	HSIC perm	HSIC gamma	SNR_1	SNR_2
1	0.0099	0	0	0	0.861	0.821
1/2	0.0099	0	0	0	0.434	0.416
1/4	0.0099	0	0	0	0.221	0.211
1/8	0.0396	0.0320	0.0375	0.0381	0.118	0.110
1/16	0.2473	0.2635	0.2518	0.2492	0.071	0.063
1/32	0.4073	0.4200	0.4102	0.3992	0.055	0.048

Table 3.3 Relationship between independence criteria and the SNR's for the non-linear relationship with varying noise levels.

Sinusoidal relationship

We continue by investigating the non-linear relationship between variables X and Y and its effect on the SNR and the independence criteria. We chose the sinusoidal relationship:

$$\begin{aligned} X &\approx U[0, 10] \\ Y &\approx S\sin(X) + U[0, 1] \end{aligned} \tag{3.7.4}$$

As in the linear case, the signal varies from 1 to 2^{-5}, i.e. $S \in \{1, 2^{-1}, ..., 2^{-5}\}$. Empirical results are provided in Table 3.3. We observe the same behaviour, at the threshold of approximately 0.1 the statistical independence tests cannot reject independence any more.

Empirical motivation

While exploring the single-cell datasets from Sachs et al. (2005) we noticed that some edges are found easily while kernel PC (for full discussion see Chapter 4 Section 4) fails to find others, which are suppose to be there based on the golden standard. The reason becomes evident if we look at the datasets from an SNR point of view. Consider the matrix of SNR values for all the variables regressed on each other. We would not expect to find an edge between two variables if the SNR is below the threshold in both directions. On the other hand, if the SNR is large it does not necessarily imply an edge between the variables as their dependence might be a result of latent variables. That is, $SNR(x, y) >$ "threshold" can be caused by:

- $x \to y$
- $x \leftarrow y$
- $x \leftarrow z \to y$
- $x \to z \to y$
- $x \leftarrow z \to y$

Only in the first two cases would we find an edge between x and y. In the other cases x and y are independent conditioned on z. Even given this consideration, we can treat this SNR matrix as an "upper bound" on what graph structure we may expect to find. Tables 3.4 and 3.5 are comprised of the SNR_1 and SNR_2 values, where the value in the i^{th} row and j^{th} column is the SNR for X_j regressed on X_i. Values greater than the thresholds determined above (the p-value for the independence criteria based test is < 0.05) are in red. We note that both approaches yield almost identical result. Values in bold are the ones where we expect to find a dependency. Based on the SNR matrix we conclude that 7 of the well known dependencies are extremely unlikely to be detected even in the generated data model:

- pKA $\to$ pRAF
- PKC $\to$ pRAF
- pMEK $\to$ ERK
- PIP3 $\to$ AKT
- pLC$_\gamma$ $\to$ PKC
- pKA $\to$ P38
- PKC $\to$ pJNK

Edge PKC $\to$ PIP2 is right on the verge of the threshold in both SNR approaches. We expect to detect the remaining 8 well known dependencies based on their large SNR:

- pRAF → pMEK
- PIP3 → PIP2
- PKC → P38
- pLC_γ → PIP2
- pKA → ERK
- PIP3 → pLC_γ
- pKA → AKT
- PKC → pJNK

Based on the SNR approach we conclude that we would expect to have:

- True positive: $8-9$
- True negative: 37
- False positive: 2
- False negative: $7-8$

This result agrees with our simulations.
As the SNR is much quicker to calculate and it gives results very similar to the ones produced using the independence criteria it might be advantageous to explore this path more.

	pRAF	pMEK	pLC_γ	PIP2	PIP3	ERK	AKT	pKA	PKC	p38	pJNK
pRAF	0.00	**12.19**	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00
pMEK	11.66	0.00	0.00	0.00	0.00	**0.00**	0.00	0.00	0.00	0.00	0.00
pLC_γ	0.00	0.00	0.00	**3.91**	0.21	0.00	0.00	0.00	**0.00**	0.00	0.00
PIP2	0.00	0.00	3.93	0.00	0.53	0.00	0.00	0.00	0.01	0.00	0.00
PIP3	0.00	0.00	**0.28**	**0.52**	0.00	0.00	**0.00**	0.01	0.00	0.00	0.00
ERK	0.00	0.00	0.00	0.00	0.00	0.00	3.86	0.51	0.00	0.00	0.00
AKT	0.00	0.00	0.00	0.00	0.00	4.43	0.00	0.65	0.00	0.00	0.00
pKA	**0.00**	0.00	0.00	0.00	0.00	**0.52**	**0.54**	0.00	0.00	**0.00**	**0.00**
PKC	**0.00**	0.00	0.00	**0.01**	0.00	0.00	0.00	0.00	0.00	**7.39**	**0.55**
P38	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	7.01	0.00	0.41
pJNK	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.46	0.34	0.00

Table 3.4 SNR_1 for all 8 datasets combined.

The following Figure 3.10 shows the graph structure and the edges of high (green) and low (gray) SNR.

3.7.3 Signal to Noise Ratio Independence Criterion

The results from Section 3.7.2 give us the idea that SNR can be used to establish a measurement of independence between two variables. To transform this idea into a working criterion we start by defining the relationship between the SNR and the

	pRAF	pMEK	pLC$_\gamma$	PIP2	PIP3	ERK	AKT	pKA	PKC	p38	pJNK
pRAF	0.00	**1.46**	0.04	0.04	0.07	0.05	0.06	0.05	0.04	0.03	0.04
pMEK	1.51	0.00	0.03	0.04	0.06	**0.06**	0.04	0.03	0.03	0.04	0.02
pLC$_\gamma$	0.07	0.08	0.00	**0.76**	0.31	0.05	0.06	0.09	**0.02**	0.04	0.02
PIP2	0.08	0.03	0.73	0.00	0.58	0.05	0.05	0.07	0.06	0.05	0.03
PIP3	0.08	0.06	**0.38**	**0.47**	0.00	0.03	**0.03**	0.10	0.04	0.07	0.02
ERK	0.03	0.02	0.09	0.04	0.06	0.00	1.75	0.71	0.02	0.08	0.03
AKT	0.03	0.03	0.04	0.03	0.07	1.41	0.00	0.75	0.02	0.03	0.02
pKA	**0.02**	0.03	0.03	0.02	0.04	**0.56**	**0.69**	0.00	0.02	**0.04**	**0.03**
PKC	**0.07**	0.05	0.06	**0.11**	0.05	0.06	0.06	0.12	0.00	**1.17**	**0.44**
P38	0.06	0.04	0.04	0.03	0.04	0.06	0.07	0.09	0.97	0.00	0.42
pJNK	0.09	0.05	0.05	0.03	0.04	0.02	0.05	0.04	0.43	0.53	0.00

Table 3.5 SNR_2 for all 8 datasets combined.

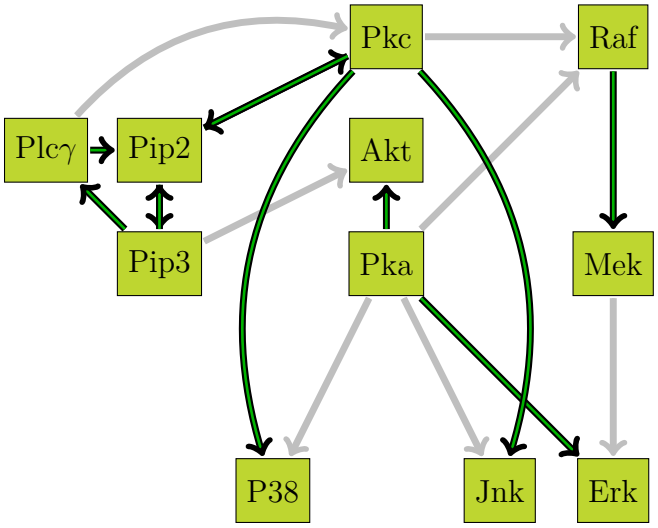


Fig. 3.10 Summary of known dependencies with high (green) and low (gray) SNR.

general independence, then we provide some theoretical results and finally provide empirical estimates of its performance with respect to other independence criteria on simulated examples.

3.7.4 SNRIC and the General Independence

Recall that random variables X and Y are independent if and only if their joint probability density function factorizes to the product of their marginal pdfs, i.e. $f_{X,Y} = f_X f_Y$. An alternative way of defining the independence between two random variables is using moment generating functions.

Theorem 3.7.1. Suppose that X and Y are random variables with the moment generating functions M_X and M_Y respectively and the joint moment generating function $M_{X,Y}$, then X and Y are independent if and only if

$$M_{X,Y}(s,t) = M_X(s)M_Y(t) \tag{3.7.5}$$

These definitions are equivalent as the second equality in the equation below is true if and only if $f_{X,Y} = f_X f_Y$.

$$M_{X,Y}(s,t) = \int e^{sX+tY} f_{X,Y}(x,y)dxdy = \int e^{sX} f_X(x)dx \int e^{tY} f_Y(y)dy = M_X(s)M_Y(t)$$

Proposition 3.7.2. Random variables X and Y with the support $\mathcal{X}$ are independent iff

$$\mathrm{E}_Y[Y^n \mid X = x] = \mathrm{E}_Y[Y^n],\ \forall x \in \chi \text{ and } \forall n = 1, 2, ... \tag{3.7.6}$$

Proof. Write the moment generating functions as $M_X(s) = \mathrm{E}_X[e^{sX}]$, $M_Y(t) = \mathrm{E}_Y[e^{tY}]$ and $M_{X,Y}(s,t) = \mathrm{E}_X[e^{sX+tY}]$. It follows that

$$\begin{aligned}
\mathrm{E}_{X,Y}[e^{sX+tY}] &= \mathrm{E}_X\Big[e^{sX}\mathrm{E}_{Y|X}[e^{tY} \mid X]\Big] \\
&= \mathrm{E}_X\left[e^{sX}\mathrm{E}_{Y|X}\Big[\sum_{n=0}^{\infty}\frac{t^n}{n!}Y^n \mid X\Big]\right] \\
&= \mathrm{E}_X\Big[e^{sX}\sum_{n=0}^{\infty}\frac{t^n}{n!}\mathrm{E}_{Y|X}[Y^n \mid X]\Big] \\
&= \mathrm{E}_X\Big[e^{sX}\sum_{n=0}^{\infty}\frac{t^n}{n!}\mathrm{E}_Y[Y^n]\Big] \\
&= \mathrm{E}_X[e^{sX}]\mathrm{E}_Y[e^{tY}]
\end{aligned} \tag{3.7.7}$$

It follows from Theorem 3.7.1 that X and Y are independent.

□

To make this work we need to estimate $\mu^k_{Y|x} = \mathrm{E}[Y^k|X = x]$ for any k and all x. We only have a finite sample $(x, y) = ((x_1, y_1), ..., (x_n, y_n))$. We define an empirical estimate of $\mu^k_{Y|x}$ as $\hat{\mu}^k_{Y|i}$:

$$\mu^k_{Y|i}(\epsilon, n) = \frac{1}{n_i(n,\epsilon)} \sum_{j:\|x_i - x_j\|<\epsilon} y_j^k \tag{3.7.8}$$

where

$$n_i(n,\epsilon) = \sum_{j=1}^{n} \mathbb{1}(j : \|x_i - x_j\| < \epsilon)$$

is the size of an ϵ radius ball centred at x_i, i.e. the number of observations in this ball. Taking the limits in a correct order yields:

$$\lim_{\epsilon \to 0}(\lim_{n \to \infty} \mu^k_{Y|i}(\epsilon, n)) = \mathrm{E}[Y^k | X = x_i] \tag{3.7.9}$$

therefore $\hat{\mu}^k_{Y|i}$ is an asymptotically correct estimate of $\mu^k_{Y|x}$.

3.7.5 Definition of the SNRIC

To establish a general independence criterion we would need to calculate all the moments of Y conditioned on $X = x$ (finding $\mathrm{E}[Y^k|X = x]$ for all $k = 1, 2, ...$ and $x \in \mathcal{X}$), but that is not feasible.

We hope that it is sufficient to calculate only a few of the conditional moments to infer some knowledge about the dependence between the random variables X and Y. As it turns out, if we assume a specific functional form (namely an additive and multiplicative noise model) between the random variables, it is enough to calculate only the first two conditional moments. This turns out to be sufficient for most of the practical implications.

We also need to choose the regression method. We chose to use cubic regression. This was done due to the time constraint of using SNRIC in long MCMC simulations done in Chapter 5. We show in Section 3.7.7 that cubic regression works well on a wide range of simulated examples and therefore is a good compromise between time efficiency and accuracy of the method. Future work is to extend SNRIC to use a more sophisticated regression method, for example splines from `R` package `mgcv` like we did in the conditional independence test using residuals (see Section 3.6.3). This method is more flexible and has in-built protection from over-fitting but is too slow to be used in the current MCMC framework (for more detail see Section 5.6.2).

Algorithm 3 shows how to calculate the empirical estimate of the SNRIC.

In this algorithm we are following the first interpretation of SNR, i.e. $\mathrm{SNR} = \sqrt{\frac{\mathrm{Var(Signal)}}{\mathrm{Var(noise)}}}$. SNR_μ can be interpreted as the signal to noise ratio of $\mathrm{E}[Y \mid X = x] = \mathrm{signal}(x) + \mathrm{noise}$ and SNR_V can be interpreted as the signal to noise ratio of $\mathrm{Var}[Y \mid X = x] = \mathrm{signal}(x) + \mathrm{noise}$ (it is equivalent to $\mathrm{E}[Y^2 \mid X = x]$ but provides better interpretability).

3.7.6 SNR Independence Test

Similarly as in the kernel independence criterion (such as HSIC or dCov) case, SNRIC provides us only with a number which is not necessarily meaningful without a context.

Data: x, y
Result: $\mathrm{SNR}(y, x)$ the empirical estimate of the SNRIC
Regress y on x to obtain $y = f_1(x) + \epsilon_1$
Calculate $\mathrm{SNR}_\mu(y, x) = \sqrt{\frac{\mathrm{Var}(f_1(x))}{\mathrm{Var}(\epsilon_1)}}$
Calculate the "second moment estimate" of Y: $\widehat{\mathrm{Var}[y]} = (y - f_1(x))^2$
Regress $\widehat{\mathrm{Var}[y]}$ on x to obtain $\widehat{\mathrm{Var}[y]} = f_2(x) + \epsilon_2$
Calculate $\mathrm{SNR}_V(y, x) = \sqrt{\frac{\mathrm{Var}(f_2(x))}{\mathrm{Var}(\epsilon_2)}}$
$\mathrm{SNR}(y, x) = \max\left(\mathrm{SNR}_\mu(y, x), \mathrm{SNR}_V(y, x)\right)$
return $\mathrm{SNR}(y, x)$

Algorithm 3: Empirical estimate of the SNRIC

Note that SNRIC can be used on its own in the MCMC sampling scheme, where our aim would be to minimize it across all variable pairs. In the context of the PC algorithm we are trying to answer the question whether two random variables are independent or not. In order to do so we require a test that will provide us with a p-value (the probability to observe a sample at least as dependent as our sample, given that the two random variables are actually independent). We provide two approaches inspired by the kernel independence tests, namely the permutation test and the Gamma test.

Permutation test

Permutation test works in exactly the same way as the HSIC or dCov permutation tests. This approach requires to recalculate the independence criteria many times, so might be time consuming. In order to reduce some of the calculations we may reuse some of the previous results. We are using a linear regression in our implementation.

Gamma test

The alternative is to approximate the $SNR(X, Y)$ distribution under the null hypothesis (X and Y are independent) by a Gamma distribution. We currently do not have a closed form for this distribution. But it is worth noting that the Gamma approximation is much more affected by the number of points in the sample, rather than by the actual distribution of the random variables themselves. We can see this in Figure 3.11. For the sample sizes of 100, 300, 500 and 1000 we plotted the cumulative distribution function for $SNR(X, Y)$ where X and Y are independent random variables sampled from the combination of distributions provided in Table 3.6. We observe that for the sample sizes of at least 300 all the CDF's are almost identical and are well approximated by

the Gamma distribution. This suggests, that we can approximate the $SNR(X, Y)$ IC under the null hypothesis by a Gamma distribution for any underlying real distributions of X and Y, based on the sample size alone.

X	Y
Uniform(0,1)	Uniform(0,1)
Uniform(0,1)	Normal(0,1)
Normal(0,1)	Normal(0,1)
Normal(0,1)	Gamma(1,1)
Gamma(1,1)	Gamma(500,0.2)
Gamma(1,1)	Mixture of N(-5,1) and N(5,1)

Table 3.6 Combinations of distributions used to generate Figure 3.11.

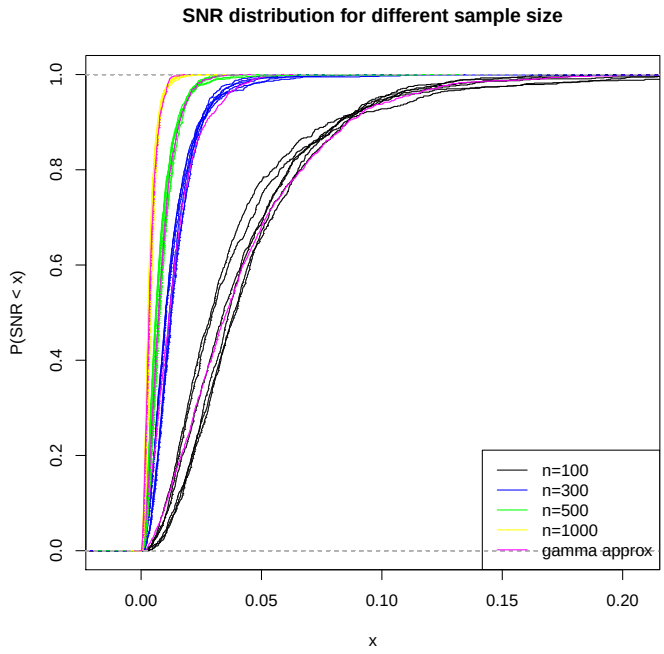


Fig. 3.11 CDF for SNR approximation using a Gamma distribution

3.7.7 Unconditional Independence Test Examples

In this section we explore the effectiveness of the SNR (permutation and Gamma) tests in comparison with the well established general independence tests based on the distance

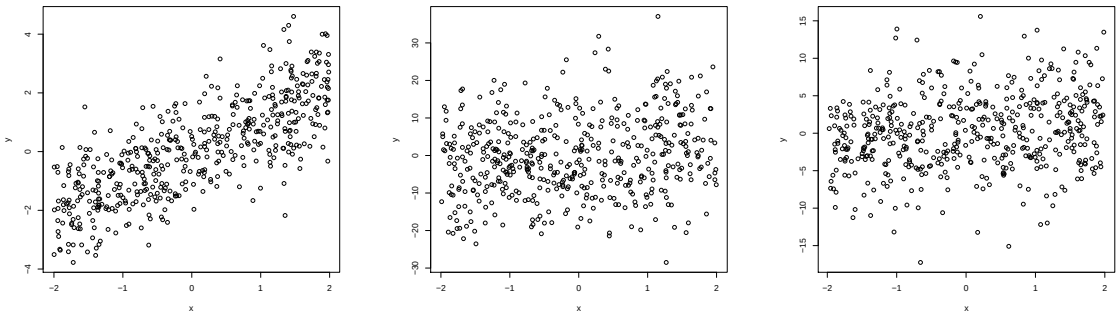
covariance (using the dCov permutation variant) and the Hilbert Schmidt Independence Criteria (using the HSIC gamma variant). To do so we use five different dependence structures commonly used in literature: linear dependence, non-linear dependence, rotation of independent data, latent dependence and finally the heteroscedastic data.

Linear dependency

Linear dependency model:

$$\begin{aligned} X &\sim U[-2, 2], \\ Y &\sim X + N(0, \sigma^2) \end{aligned} \tag{3.7.10}$$

The example of the dataset generated using this model is provided in Figure 3.12. In Table 3.7 we provide the p-values of four independence tests applied to the datasets generated from this model. We are using the SNRIC permutation and gamma, dCov permutation and HSIC gamma tests. Each value in the table is the mean of each test applied to 100 simulated datasets from the linear dependency model. The results between the SNRIC permutation, SNRIC gamma and dCov permutation tests are very close, the HSIC gamma test seems to perform similarly, but slightly worse in this case.



(a) $\epsilon \sim N(0, 1)$. (b) $\epsilon \sim N(0, 25)$. (c) $\epsilon \sim N(0, 100)$.

Fig. 3.12 Three examples of data with linear dependency and varying noise.

Non-linear dependency

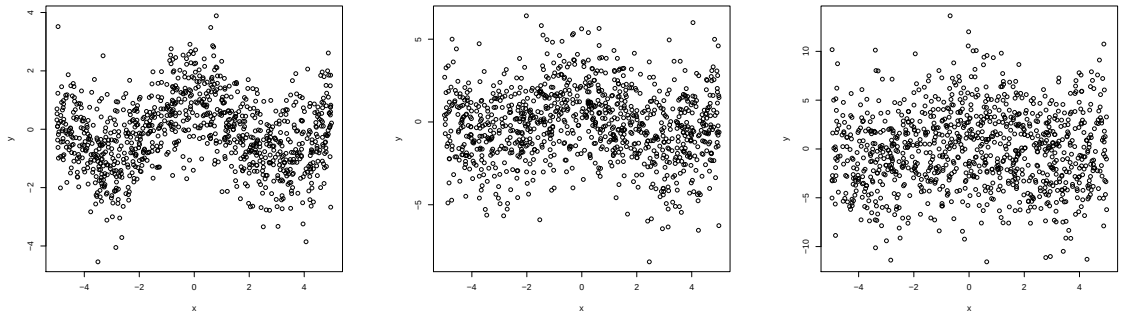
Non-linear dependency model:

$$\begin{aligned} X &\sim U[-2, 2], \\ Y &\sim \cos(X) + N(0, \sigma^2) \end{aligned} \tag{3.7.11}$$

Test	$\sigma = 1$	$\sigma = 2$	$\sigma = 3$	$\sigma = 5$	$\sigma = 10$	$\sigma = 100$
SNR perm	0.01	0.01	0.01	0.01	0.17	0.48
SNR gamma	0.00	0.00	0.00	0.00	0.14	0.40
dCov perm	0.01	0.01	0.01	0.01	0.15	0.53
HSIC gamma	0.00	0.00	0.00	0.06	0.56	0.58

Table 3.7 The p-value estimates on the data with linear dependence and varying noise.

The example of the datasets generated using this model is provided in Figure 3.13. In Table 3.8 we provide the p-values of four independence tests. The set up for generating the table is the same as in the linear dependence case. Results produced by all four tests are again very similar with the SNRIC based tests being slightly worse. This can be explained by the fact that the SNRIC tests are taking into account only up to the cubic term in regressing Y on X, it does not give us a perfect regression in this case. We should note that the number of power terms taken in the regression can be manually tuned for any network inference problem if we believe that it is required. The time efficiency loss will not be too large as the leading term in the linear regression will be the number of sample points, not the number of covariates.



(a) $\epsilon \sim N(0, 1)$. (b) $\epsilon \sim N(0, 4)$. (c) $\epsilon \sim N(0, 9)$.

Fig. 3.13 Three examples of data with nonlinear dependency and varying noise.

Latent dependency

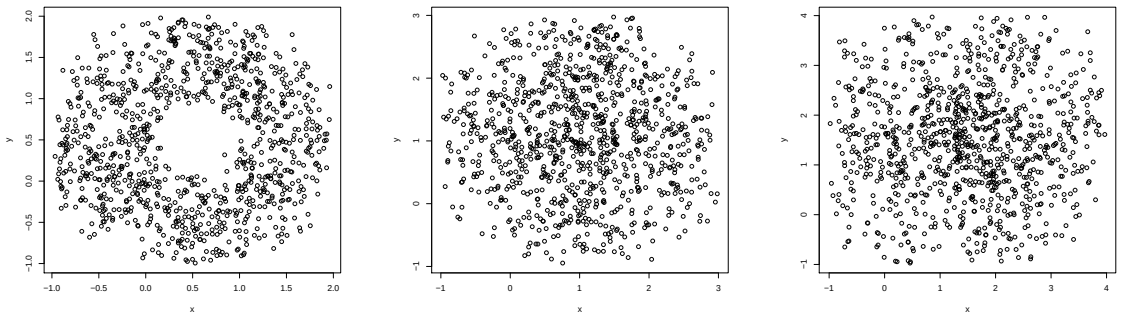
Latent dependency model:

$$\begin{aligned} \Theta &\sim U[-\pi, \pi], \\ X &\sim \cos(\Theta) + U[-\epsilon/2, \epsilon/2], \\ Y &\sim \sin(\Theta) + U[-\epsilon/2, \epsilon/2] \end{aligned} \tag{3.7.12}$$

Test	$\sigma = 1$	$\sigma = 1.5$	$\sigma = 2$	$\sigma = 2.5$	$\sigma = 3$	$\sigma = 4$
SNRIC perm	0.01	0.02	0.08	0.14	0.23	0.33
SNRIC gamma	0.00	0.01	0.05	0.10	0.18	0.27
dCov perm	0.01	0.01	0.01	0.02	0.06	0.14
HSIC gamma	0.00	0.00	0.01	0.03	0.12	0.24

Table 3.8 The p-value estimates on the data with the non-linear dependence and varying noise.

The example of the datasets generated using this model is provided in Figure 3.14. In Table 3.9 we provide the p-values of four independence tests. The set up for generating the table is the same as in the linear dependence case. SNRIC based tests produce better results in this case: they still find dependence with the noise level $\sigma = 2$, while dCov and HSIC based tests declare independence. This can be explained by the fact that the SNRIC is specifically designed to catch the varying noise levels of Y with respect to X ($\mathrm{Var}[Y \mid X = x]$ for varying x), while the dCov and HSIC are more general independence tests.



(a) $\epsilon \sim U[0, 1]$. (b) $\epsilon \sim U[0, 2]$. (c) $\epsilon \sim U[0, 3]$.

Fig. 3.14 Three examples of data with circular dependency and varying noise.

Test	$\sigma = 1$	$\sigma = 2$	$\sigma = 2.2$	$\sigma = 2.5$	$\sigma = 3$	$\sigma = 3.5$
SNRIC perm	0.01	0.09	0.13	0.27	0.43	0.47
SNRIC gamma	0.00	0.07	0.10	0.22	0.35	0.40
dCov perm	0.01	0.37	0.37	0.44	0.48	0.48
HSIC gamma	0.00	0.28	0.31	0.42	0.48	0.47

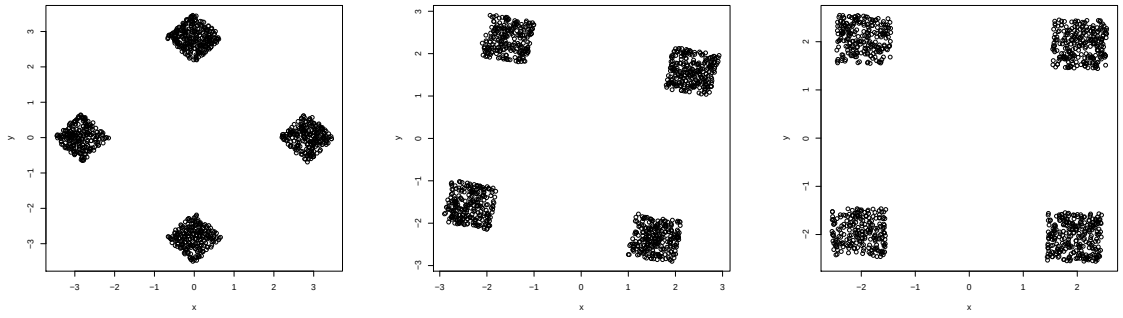
Table 3.9 The p-value estimates on the circular dependency data with varying noise.

Rotational dependency

Rotational dependency model:

$$\begin{aligned} Z &\sim U[-0.5, 0.5] + 4\,\text{Bern}(0.5) - 2, \\ W &\sim U[-0.5, 0.5] + 4\,\text{Bern}(0.5) - 2, \\ X &\sim \cos(\theta) \times Z + \sin(\theta) \times W, \\ Y &\sim -\sin(\theta) \times Z + \cos(\theta) \times W \end{aligned} \tag{3.7.13}$$

The example of the dataset generated using this model is provided in Figure 3.15. In Table 3.10 we provide the p-values of four independence tests. The set up for generating the table is the same as in the linear dependence case. All four tests performed similarly well in this case.



(a) $\theta = \pi/4$. (b) $\theta = \pi/16$. (c) $\theta = \pi/128$.

Fig. 3.15 Three examples of independent data after varying rotations applied.

Test	$\theta = \pi/4$	$\theta = \pi/8$	$\theta = \pi/16$	$\theta = \pi/32$	$\theta = \pi/64$	$\theta = \pi/128$
SNRIC perm	0.01	0.01	0.01	0.01	0.01	0.04
SNRIC gamma	0.00	0.00	0.00	0.00	0.00	0.02
dCov perm	0.01	0.01	0.01	0.01	0.01	0.09
HSIC perm	0.00	0.00	0.00	0.00	0.00	0.06

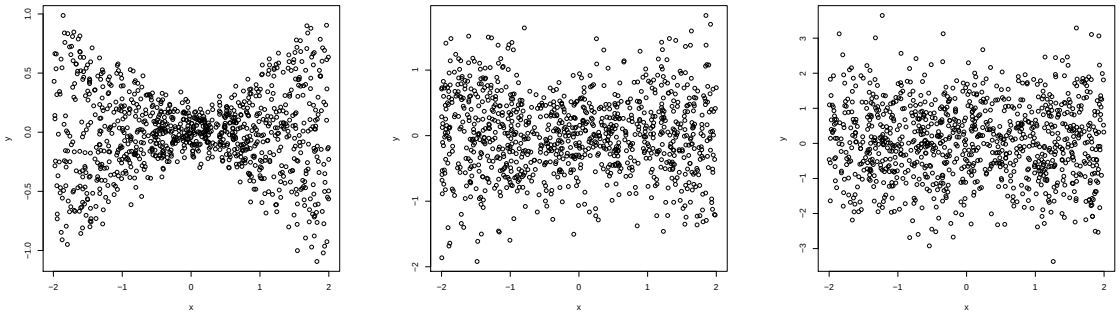
Table 3.10 The p-value estimates on the independent data after varying rotation.

Heteroscedastic data

Noise level dependency model:

$$\begin{aligned} X &\sim U[-2,2], \\ Y &\sim X \times U[-0.5,0.5] + N(0,\sigma^2) \end{aligned} \tag{3.7.14}$$

The example of the datasets generated using this model is provided in Figure 3.16. In Table 3.11 we provide the p-values of four independence tests. The set up for generating the table is the same as in the linear dependence case. SNRIC based tests produce better results in this case: they still find dependence with the noise level $\sigma = 0.8$, while the dCov and HSIC based tests declare independence. This can be explained in the same way as in the latent dependency case.



(a) $\epsilon \sim N(0, 0.01)$. (b) $\epsilon \sim N(0, 0.25)$. (c) $\epsilon \sim N(0, 1)$.

Fig. 3.16 Three examples of data with the noise level dependency and varying noise.

Test	$\sigma = 0.5$	$\sigma = 0.6$	$\sigma = 0.7$	$\sigma = 0.8$	$\sigma = 0.9$	$\sigma = 1$
SNRIC perm	0.01	0.01	0.01	0.02	0.28	0.31
SNRIC gamma	0.00	0.00	0.00	0.01	0.23	0.25
dCov perm	0.01	0.01	0.06	0.18	0.38	0.44
HSIC gamma	0.00	0.00	0.02	0.09	0.35	0.42

Table 3.11 The p-value estimates on the data with the noise level dependency and varying noise.

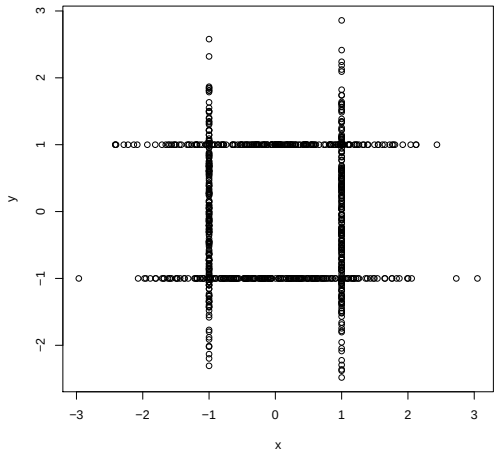
Conclusions

We note that both the SNRIC permutation and Gamma tests produced very similar p-values in all five cases, this acts as the reassurance that using the estimated parameters for the Gamma approximation works well in practice. Also we should note that the p-values produced by the SNRIC tests are overall very similar to the ones of the

dCov and HSIC tests. SNRIC tests outperformed the general independence tests in the cases where dependency comes from varying levels of noise (circular and varying noise models) and were slightly worse in the non-linear dependency model, where the polynomial regression in the SNRIC failed to regress data correctly. This can be easily fixed by using either more power terms in the polynomial regression or use a better regression method (for example non-linear regression with splines), but this would reduce computational speed.

3.7.8 Counter Example to SNRIC

So far we have shown that if data comes from the additive noise model SNRIC based test perform almost as well as the general independence criteria based tests. On the other hand if data does not come from an additive noise model there is no guarantee that SNRIC would be able to find dependency. Consider an example from Figure 3.17.



(a) Scatterplot.

$$
\begin{aligned}
Z &\sim \text{Bern}(0.5) \\
X \mid Z = 0 &\sim N(0,1), \\
Y \mid Z = 0 &\sim 2\text{Bern}(0.5) - 1, \\
X \mid Z = 1 &\sim 2\text{Bern}(0.5) - 1, \\
Y \mid Z = 1 &\sim N(0,1)
\end{aligned}
$$

(b) Distribution.

Fig. 3.17 Counterexample for SNRIC.

In this case $\mathrm{E}[X \mid Y = y] = 0$ and $\mathrm{E}[X^2 \mid Y = y] = 1$ for all $y \in \mathbb{R}$. Similarly $\mathrm{E}[Y \mid X = x] = 0$ and $\mathrm{E}[Y^2 \mid X = x] = 1$ for all $x \in \mathbb{R}$. SNRIC is considering only the first two conditional moments therefore it is unable to detect the dependence between X and Y. The general independence criteria find clear dependence between S and Y, see Table 3.12.

Test	SNRIC perm	SNRIC gamma	dCov perm	HSIC gamma
p-value	0.57	0.56	<0.01	0.07

Table 3.12 The p-value estimates on the counter example for SNRIC data.

3.7.9 SNRIC Conditional Independence Test

We have presented the SNRIC based independence test and discussed its performance in comparison to the dCov and HSIC based independence tests. We should note that having only the unconditional variant of the independence test is not sufficient for applying this criterion in the PC algorithm framework. Therefore in this section we introduce the SNRIC conditional independence test and discuss its performance with respect to other options such as dCov-resid, HSIC-resid and HSIC-clust.
The SNR conditional independence test uses a similar approach to the dCov/HSIC residuals conditional independence tests. Given observations from the random variables X, Y and Z we want to check if X and Y are independent given Z. Suppose we have m triples of observations $\{(x_1, y_1, z_1), ..., (x_m, y_m, z_m)\}$.

Data: x, y, z, ϵ
Result: p-value
Regress observations x on z to obtain the conditional mean estimates $m_x(i)$;
Regress observations y on z to obtain the conditional mean estimates $m_y(i)$;
Regress expected variance $(x - m_x)^2$ on z to obtain the conditional variance estimates $V_x(i)$;
Regress expected variance $(y - m_y)^2$ on z to obtain the conditional variance estimates $V_y(i)$;
Calculate the residuals of x given z as $r_x(i) = \frac{x_i - m_x(i)}{\max(\sqrt{V_x(i)}, \epsilon)}$;
Calculate the residuals of y given z as $r_y(i) = \frac{y_i - m_y(i)}{\max(\sqrt{V_y(i)}, \epsilon)}$;
Calculate the p-value p for $SNR(r_x, r_y)$;
return p

Algorithm 4: SNR conditional independence test

The algorithm depends on an additional regularization parameter ϵ, which is required for the stability of the algorithm. The predicted variance could be arbitrarily close to zero and that would result in a very large variability of the residuals.

3.7.10 Proofs

As discussed above the $SNR(X, Y)$ being equal to zero does not mean that the underlying random variables X and Y are truly independent. Though in some special cases where we assume specific relationship between X and Y SNRIC being zero is equivalent to X and Y being independent. In this subsection we provide some theoretical results on when the SNRIC being zero implies independence.

Theorem 3.7.3. Assume Y follows an additive and multiplicative noise model, that is

$$Y \mid x \sim h(x, U, V) = f(x) + U \times g(x) + V, \tag{3.7.15}$$

where U and V are independent random variables and f and g are piece-wise continuous functions. If the conditions 3.7.16 and 3.7.17 are met, then $Y = h(V)$, i.e. Y is a function of V only and is independent of X.

$$\mathrm{E}\Big[h(x, U, V) \mid x = x_1\Big] = \mathrm{E}\Big[h(x, U, V) \mid x = x_2\Big], \text{ for all } x_1, x_2 \in \mathcal{X} \tag{3.7.16}$$

$$\mathrm{E}\Big[h^2(x, U, V) \mid x = x_1\Big] = \mathrm{E}\Big[h^2(x, U, V) \mid x = x_2\Big], \text{ for all } x_1, x_2 \in \mathcal{X} \tag{3.7.17}$$

Proof. Suppose the condition 3.7.16 holds, i.e. for all $x_1, x_2 \in \mathcal{X}$

$$\begin{aligned} 0 &= \mathrm{E}\Big[h(x, U, V) \mid x = x_1\Big] - \mathrm{E}\Big[h(x, U, V) \mid x = x_2\Big] \\ &= f(x_1) + \mu_U g(x_1) + \mu_v - f(x_2) - \mu_U g(x_2) - \mu_V \\ &= f(x_1) - f(x_2) + \mu_U\Big(g(x_1) - g(x_2)\Big) \end{aligned} \tag{3.7.18}$$

then it follows that

$$f(x_1) + \mu_U g(x_1) = f(x_2) + \mu_U g(x_2), \ \forall x_1, x_2 \in \mathcal{X} \tag{3.7.19}$$

this is equivalent to

$$f(x) + \mu_U g(x) = C, \text{ for some constant } C \text{ and } \forall x \in \mathcal{X} \tag{3.7.20}$$

this is equivalent to

$$f(x) = -\mu_U g(x) + C, \text{ for some constant } C \text{ and } \forall x \in \mathcal{X} \tag{3.7.21}$$

Now we can rewrite h as

$$Y = h(x, U, V) = (U - \mu_U)g(x) + C + V \tag{3.7.22}$$

Suppose the condition 3.7.17 holds, i.e. for all $x_1, x_2 \in \mathcal{X}$

$$\begin{aligned}
0 &= \mathrm{E}\Big[h^2(x,U,V) \mid x = x_1\Big] - \mathrm{E}\Big[h^2(x,U,V) \mid x = x_2\Big] \\
&= \mathrm{Var}\Big[h(x,U,V) \mid x = x_1\Big] - \mathrm{Var}\Big[h(x,U,V) \mid x = x_2\Big] \\
&= \mathrm{Cov}((U-\mu_U)g(x_1) + C + V, (U-\mu_U)g(x_1) + C + V) \\
&\quad - \mathrm{Cov}((U-\mu_U)g(x_2) + C + V, (U-\mu_U)g(x_2) + C + V) \\
&= \mathrm{Var}(U)g^2(x_1) + \underbrace{\mathrm{Var}(C)}_{=0} + \mathrm{Var}(V) + 2\underbrace{\mathrm{Cov}(U-\mu_U, C)}_{=0}g(x_1) \\
&\quad + 2\underbrace{\mathrm{Cov}(V, C)}_{=0} + \mathrm{Cov}(U-\mu_U, V)g(x_1) \\
&\quad - \mathrm{Var}(U)g^2(x_2) - \underbrace{\mathrm{Var}(C)}_{=0} - \mathrm{Var}(V) - 2\underbrace{\mathrm{Cov}(U-\mu_U, C)}_{=0}g(x_2) \\
&\quad - 2\underbrace{\mathrm{Cov}(V, C)}_{=0} - \mathrm{Cov}(U-\mu_U, V)g(x_2) \\
&= \mathrm{Var}(U)(g^2(x_1) - g^2(x_2)) + \mathrm{Cov}(U,V)(g(x_1) - g(x_2)) \\
&= \Big(g(x_1) - g(x_2)\Big)\Big(\mathrm{Var}(U)\Big(g(x_1) + g(x_2)\Big) + \mathrm{Cov}(U,V)\Big)
\end{aligned} \tag{3.7.23}$$

Now it follows that either (a):

$$g(x_1) - g(x_2) = 0, \ \forall x_1, x_2 \in \mathcal{X} \tag{3.7.24}$$

but then it follows that g is a constant function.
Or (b):

$$\mathrm{Var}(U)\Big(g(x_1) + g(x_2)\Big) + \mathrm{Cov}(U,V) = 0, \ \forall x_1, x_2 \in \mathcal{X} \tag{3.7.25}$$

We can rewrite this as

$$\mathrm{Var}(U)\Big(g(x_1) + g(x_2)\Big) = -\mathrm{Cov}(U,V), \ \forall x_1, x_2 \in \mathcal{X} \tag{3.7.26}$$

if we have x_1, x_2, x_3, such that

$$\begin{aligned}
g(x_1) + g(x_2) &\neq 0 \\
g(x_1) + g(x_3) &\neq 0 \\
g(x_2) - g(x_3) &\neq 0
\end{aligned} \tag{3.7.27}$$

then

$$\begin{aligned} 0 &= -\mathrm{Cov}(U,V) - (-\mathrm{Cov}(U,V)) \\ &= \mathrm{Var}(U)\Big(g(x_1) + g(x_2)\Big) - \mathrm{Var}(U)\Big(g(x_1) + g(x_3)\Big) \\ &= \mathrm{Var}(U)\Big(g(x_2) - g(x_3)\Big) \\ &\qquad\qquad \neq 0 \\ &\Rightarrow \mathrm{Var}(U) = 0 \Rightarrow \mathrm{Cov}(U,V) = 0 \end{aligned} \tag{3.7.28}$$

Now from both (a) and (b) it follows that

$$Y \mid x \sim \alpha_1 + \alpha_2 U + V, \text{ for some constants } \alpha_1, \alpha_2 \tag{3.7.29}$$

That is Y is independent of X.

□

The converse follows immediately from the Proposition 3.7.2 if X and Y are independent, then $\mathrm{E}[Y^n \mid X = x_1] = \mathrm{E}[Y^n] = \mathrm{E}[Y^n \mid X = x_2]$.

3.8 Conclusions

In this chapter we have presented three independence criteria from the literature. Original papers provided only the theoretical justification of the criteria. We provided the geometrical intuition behind these criteria. Sejdinovic et al. (2013) showed the equivalence between the distance based and the RKHS based independence criteria. We showed the equivalence in the case of the most popular independence criteria such as dCov, KGV and HSIC explicitly. dCov and HSIC has parameters that have to be set correctly in order to use them successfully. We have explored the space of parameters for different number of observations, varying noise levels and different dependence structures and provided suggestions for the appropriate ranges for these parameters. Finally we introduced a new approach to test for the dependence between variables, the SNRIC. We showed that at least with a simulated data it is just as effective as the general independence criteria while being significantly faster.

All statistical tests based on dCov, HSIC and SNRIC did a very good job in determining whether the samples are independent, with dCov and HSIC based tests performing slightly better. Though it is important to note that for dCov and HSIC the computational complexity of the criteria is quadratic in the number of observations, so they become quite slow to use on data with a large number of samples. Under the current implementation the computational complexity of SNRIC is linear in the number of observations.

In the following two chapters we will use these independence criteria for the network inference. First in the greedy search style algorithm like PC in Section 4 and finally in a full MCMC sampler in Section 5.

3.8.1 Future Work

1. We showed that dCov is a special case of HSIC with a special Euclidean distance kernel. So in the essence we were comparing HSIC with Gaussian kernel and HSIC with Euclidean distance kernel. An interesting avenue for the future work would be to test whether HSIC work better with other kernels, for example linear, polynomial, hyperbolic tangent, Laplacian or Bessel. They are conveniently implemented in `kernlab` (Karatzoglou and Smola, 2003) package for the `R` statistical environment (R Core Team, 2014).

2. Current implementation of SNRIC uses polynomial regression. This choice was made to maximize the computational efficiency (polynomial regression is fastest choice after linear regression) while preserving sufficiently good performance (SNRIC performed equally well as the general independence criteria in all cases except the periodic relationships between variables). Future extensions of the method would involve finding a better regression method that would provide better accuracy for the method while not compromising its computational efficiency. Possible approach would be to use splines (for example cubic splines, B-splines (de Boor, 1978) or P-splines (Eilers and Marx, 1996)).

Chapter 4

Independence Criteria Based PC Algorithm

4.1 Introduction

In this chapter we discuss the possibility to use independence criteria in network inference. This Chapter is joint work with Nina Desgranges.
To do so we use a variation of the popular PC algorithm. The idea was introduced in Gretton et al. (2009) and Tillman (2009). They use the Hilbert-Schmidt Independence Criterion (for full detail, see Section 3.4) for testing for independence and the cluster permutation conditional independence test (for full detail, see Section 3.6.3). We expand on this approach by adding more independence criteria, namely the distance covariance based dCov and the signal to noise ratio based SNRIC. Previously the dCov was implemented as a package in `R`, we implemented the HSIC and SNRIC as well as the adjusted PC version the kPC in the `R` package `kpcalg` (Verbyla et al., 2017).
We start this chapter by introducing the PC algorithm variation, the kPC. We continue by discussing the performance of using different independence criteria in the kPC on both simulated and experimental datasets (experimental datasets are discussed in more detail in Section 1.6).

4.2 Kernel PC

The original PC algorithm consists of the three phases described in Section 1.3: skeleton, collider and transitive. Gretton et al. (2009) presented a variation of the PC algorithm (for full detail see Section 1.3), the kernel PC or kPC. The main differences between

kPC and PC is the usage of the general independence criteria instead of a Z-test for the independence test and the generalized transitive phase instead of the transitive phase. We have discussed the general independence criteria in Section 3. Now we provide some insight into the generalized transitive phase (for full details refer to Gretton et al. (2009)).
We assume that the data comes from an additive noise model (discussed in more detail in Section 5.2.2), i.e.

$$X_i = f_i(\mathrm{Pa}_G(X_i)) + \epsilon_i, \text{ for all } i = 1, ..., m \tag{4.2.1}$$

Suppose we have an undirected edge $x - y$ with the correct directionality $x \to y$ and the true model $y = f(x) + \epsilon$, for some function f and the noise term ϵ (independent of everything else). We can regress either y on x to get the first model: $y = \hat{f}(x) + r_y$, or x on y to get the second model: $x = \hat{g}(y) + r_x$. If either f is non-linear or ϵ is non-Gaussian, then we will be able to distinguish between the two models (for more detail refer to the (Shimizu et al., 2006)).
There are three possible outcomes:

1. We get independence of the residuals in both models, i.e. $r_y \perp x$, and $r_x \perp y$. We leave the edge undirected. This could happen if the relationship f between x and y is linear and the noise is Gaussian.

2. We don't get independence of the residuals in either model, i.e. $r_y \not\perp x$, and $r_x \not\perp y$. This could happen because the data does not follow the additive noise model or there are unobserved latent variables, i.e. our model is in some way misspecified.

3. Finally, we could get independence of the residuals in one of the model, i.e. $r_y \perp x$, but not in the other, i.e. $r_x \not\perp y$. In this case we conclude that the correct orientation is $x \to y$. This is the outcome we are looking for, it happens if the data is correctly defined by the additive noise model (or at least it is sufficiently close to the true model) and we have either non-linear relationship or non-Gaussian noise.

In the generalized transitive phase we alternatively perform the above orientation algorithm and apply Meek's rules until no more orientation can be found. This approach allows us to find significantly more orientations as we show in Section 4.4.6.

4.3 Inferring Directionality Using the Independence Criteria

As we have discussed above, we may use the independence criteria to determine the direction of single edges. This is neither possible in the traditional PC algorithm nor in the discrete Bayesian Network case (as discussed in Section 2.6). We provide some simulated examples that illustrate how given that the data comes from an additive noise model the independence criteria can be used to determine the directionality of edges. We also show, how this approach may fail, if assumptions are not met, e.g. data comes from a different model or there are latent variables.

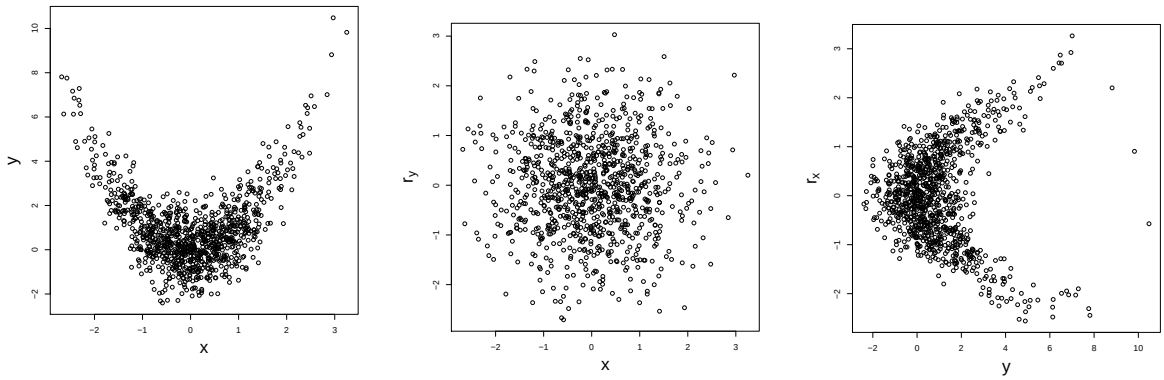
The first example is a non-linear relationship with Gaussian noise: $X \sim N(0,1)$, $Y \sim X^2 + N(0,1)$. The original dataset is shown in Figure 4.1a and Figures 4.1b and 4.1c show the residuals after regressing y on x and x on y respectively. We repeated the simulation 100 times and the resulting mean p-values of the HSIC, dCov and SNRIC tests are provided in Table 4.1. We may conclude that all tests found the residuals of y regressed on x independent of x, i.e. $r_y \perp x$, but not the other way around, i.e. $r_x \not\perp y$. Therefore in the generalized transitive phase we could safely infer $x \to y$ directionality.

Method	p-value for $r_y \perp x$	p-value for $r_x \perp y$
HSIC	0.7	0
dCov	0.86	0.01
SNRIC	0.7	0.01

Table 4.1 The mean p-values of the residuals of the regressions x on y and y on x being independent in a model with non-linear relationship with a Gaussian noise.

The second example is a linear relationship with a non-Gaussian noise: $X \sim U[0,10]$, $Y \sim 0.2X + U[0,1]$. The original dataset is shown in Figure 4.2a and Figures 4.2b and 4.2c show the residuals after regressing y on x and x on y respectively. We repeated this simulation 100 times, the resulting mean p-values of the HSIC, dCov and SNRIC tests are provided in Table 4.2. We may conclude that all tests found the residuals of y regressed on x independent of x, i.e. $r_y \perp x$, but not the other way around. Therefore in the generalized transitive phase we could safely infer $x \to y$ directionality.

The third example is a linear relationship with a Gaussian noise: $X \sim N(0,1)$, $Y \sim 0.2X + N(0,1)$. The original dataset is shown in Figure 4.3a and Figures 4.3b and 4.3c show the residuals after regressing y on x and x on y respectively. We repeated this simulation 100 times, the resulting mean p-values of the HSIC, dCov and SNRIC

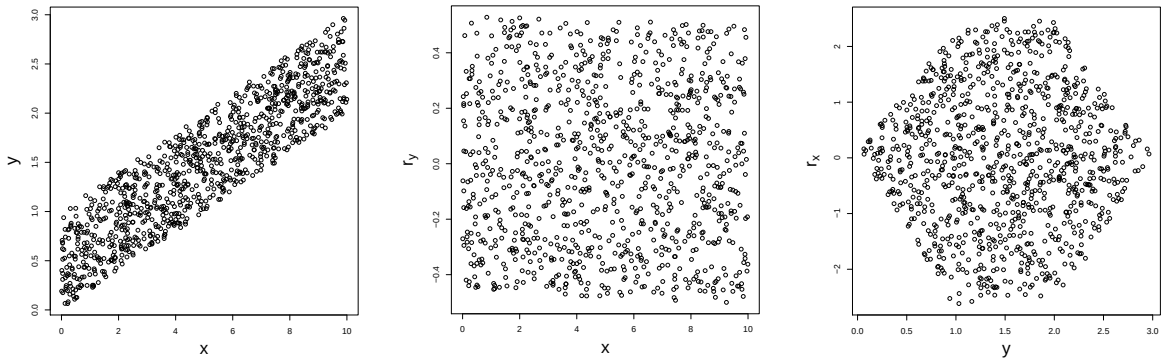


(a) True model: $y = \sin(x) + \epsilon$. (b) 1^{st} model: $y = \hat{f}(x) + r_y$. (c) 2^{nd} model: $x = \hat{g}(y) + r_x$

Fig. 4.1 Model with a non-linear relationship and Gaussian noise.

Method	p-value for $r_y \perp x$	p-value for $r_x \perp y$
HSIC	0.77	0
dCov	0.76	0.05
SNRIC	0.65	0.01

Table 4.2 The mean p-values of the residuals of the regressions x on y and y on x being independent in a model with a linear relationship with a non-Gaussian noise.



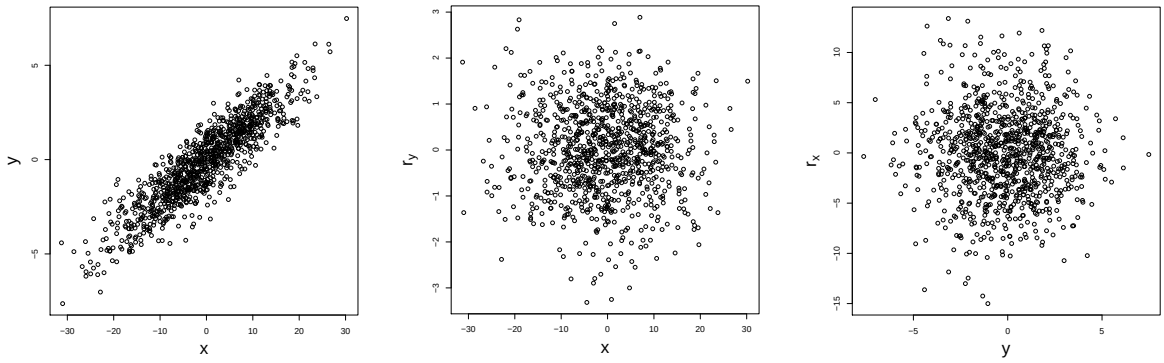
(a) True model: $y = 0.2x + \epsilon$. (b) 1^{st} model: $y = \hat{f}(x) + r_y$. (c) 2^{nd} model: $x = \hat{g}(y) + r_x$

Fig. 4.2 Model with a linear relationship and a non-Gaussian noise.

tests are provided in Table 4.3. We may conclude that all tests found both the residuals of y regressed on x independent of x, i.e. $r_y \perp x$, and the residuals of x regressed on y independent of y, i.e. $r_x \perp y$. Therefore we could not infer any directionality from such data. This is an expected result, because both models $Y \sim 0.2X + N(0, 1)$ and $X \sim 5Y + N(0, 1)$ explain the data.

Method	p-value for $r_y \perp x$	p-value for $r_x \perp y$
HSIC	0.55	0.48
dCov	0.78	0.75
SNRIC	0.65	0.64

Table 4.3 The mean p-values of the residuals of the regressions x on y and y on x being independent in a model with a linear relationship with a Gaussian noise.



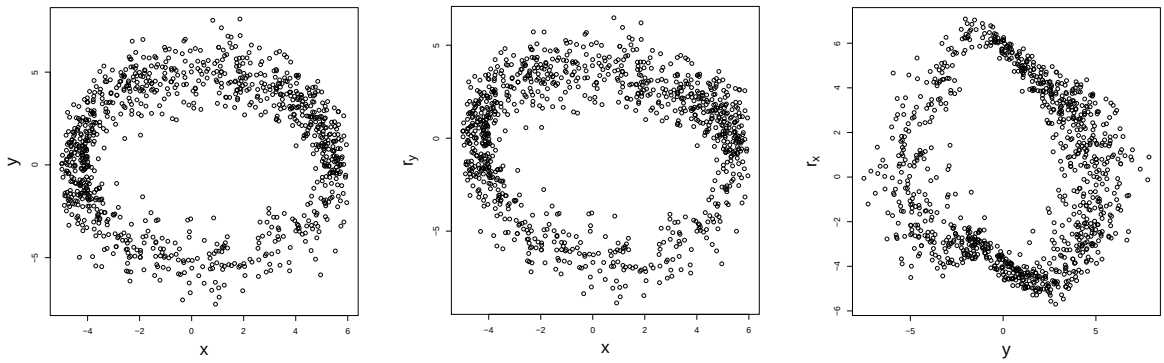
(a) True model: $y = 0.2x + \epsilon$. (b) 1^{st} model: $y = \hat{f}(x) + r_y$. (c) 2^{nd} model: $x = \hat{g}(y) + r_x$

Fig. 4.3 Model with a linear relationship and a Gaussian noise.

The fourth example is a model including a latent non-observed variable: $Z \sim U[0, 10]$, $X \sim \cos(Z) + U[0, 1]$, $Y \sim \sin(Z) + N(0, 1)$. The original dataset is shown in Figure 4.4a and Figures 4.4b and 4.4c show the residuals after regressing y on x and x on y respectively. We repeated this simulation 100 times, the resulting mean p-values of the HSIC, dCov and SNRIC tests are provided in Table 4.4. We may conclude that all tests found neither the residuals of y regressed on x independent of x, i.e. $r_y \not\perp x$, nor the residuals of x regressed on y independent of y, i.e. $r_x \not\perp y$. Therefore we could not infer any directionality from such data. This is an expected result, because neither model $x \sim f(y) + \epsilon_x$ nor $y \sim g(y) + \epsilon_y$ is the correct one. This is a good example of how unobserved latent variables might prevent correct network inference.

Method	p-value for $r_y \perp x$	p-value for $r_x \perp y$
HSIC	0.01	0
dCov	0.01	0.01
SNRIC	0.01	0.01

Table 4.4 The mean p-values of the residuals of the regressions x on y and y on x being independent in a model with an unobserved latent variable.



(a) True model: $y = 0.2x + \epsilon$. (b) 1^{st} model: $y = \hat{f}(x) + r_y$. (c) 2^{nd} model: $x = \hat{g}(y) + r_x$

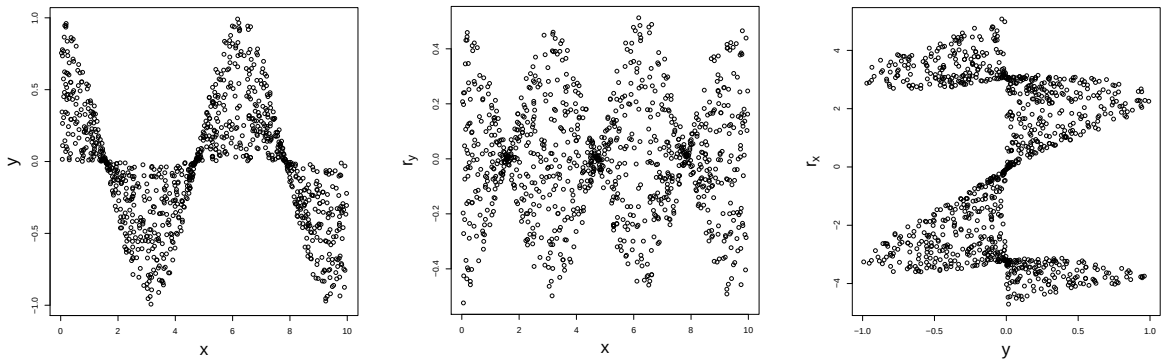
Fig. 4.4 Model with an unobserved latent variable.

The fifth and the last example is a multiplicative noise model, i.e. data does not come from an additive noise model: $X \sim U[0, 10]$, $Y \sim \cos(X) \times U[0, 1]$. The original dataset is shown in Figure 4.5a and Figures 4.5b and 4.5c show the residuals after regressing y on x and x on y respectively. We repeated this simulation 100 times, the resulting mean p-values of the HSIC, dCov and SNRIC tests are provided in Table 4.5. We may conclude that all tests found neither the residuals of y regressed on x independent of x, i.e. $r_y \not\perp x$, nor the residuals of x regressed on y independent of y, i.e. $r_x \not\perp y$. Therefore we could not infer any directionality from such data. This is an expected result, because neither model $x \sim f(y) + \epsilon_x$, nor $y \sim g(y) + \epsilon_y$ is the correct one, i.e. the true underlying model is not an additive noise model. This is a good example of how assuming a wrong model might prevent correct network inference.

Method	p-value for $r_y \perp x$	p-value for $r_x \perp y$
HSIC	0.01	0
dCov	0.01	0.01
SNRIC	0.01	0.01

Table 4.5 The mean p-values of the residuals of the regressions x on y and y on x being independent in a model that is not an additive noise model.

These five examples illustrate some of the possible situations we may encounter in the network inference while trying to direct edges. We showed that if the model is well specified and we have either non-linear relationships (1^{st} example) or the non-Gaussian noise (2^{nd} example), inferring the directionality of edges is possible. On the other hand if the model is misspecified (4^{th} and 5^{th} examples) or we are dealing with a



(a) True model: $y = 0.2x + \epsilon$. (b) 1^{st} model: $y = \hat{f}(x) + r_y$. (c) 2^{nd} model: $x = \hat{g}(y) + r_x$

Fig. 4.5 Multiplicative noise model.

truly linear relationship with a Gaussian noise (3^{rd} example) we are unable to find the directionality of the edges.

4.4 Results

We first investigate the effectiveness of the independence criteria in finding dependencies in small simulated examples. We continue with a larger scale example with data simulated by permutation resampling from the single-cell datasets. Finally we present our results for the the single-cell datasets from Sachs et al. (2005) and the Schistosomiasis dataset.

4.4.1 Comparison of Network Inference Algorithms

The performances of the algorithms are compared using ROC curves sensitivity over specificity while varying the p-value cutoff required by the test for statistical independence in the PC algorithm. Unless otherwise stated we focus on the absence and presence of edges in the inferred graphs ignoring their direction when calculating specificity and sensitivity.

Three types of datasets are considered: data simulated from a simple known network, data obtained by permuting residuals after fitting a network to single-cell data from Sachs et al. (2005), and finally the original data from the same study. Parameters of the algorithms were fixed as in Table 4.6.

For easy reference we label the algorithms as follows. The standard PC algorithm as implemented in the R package `pcalg` with the `gaussCItest` criterion (implementing Fisher's z-test for correlation) is labelled PC. The PC algorithm based on the dCov from

Table 4.6 Free parameters for kPCs and dPC.

Parameter	kPC-Clust	kPC-Resid	dPC
# of Permutations	300	300	500
Kernel width λ	1	1	NA
Regularization parameter ϵ	0.1	NA	NA
# of clusters	30	NA	NA

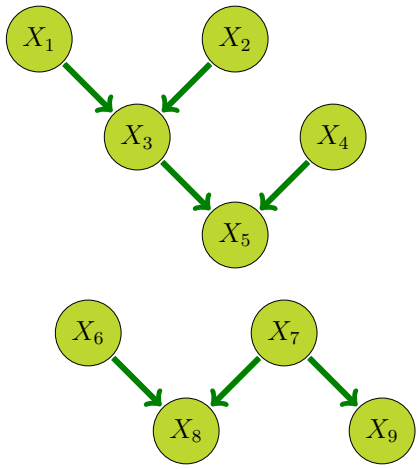
Section 3.2 is labelled dPC. The PC algorithm based on the HSIC from Section 3.4 is labelled kPC. The kPC version based on the Permutation-cluster test from Section 3.6.3 is labeled kPC-Clust. The kPC version based on the Residuals test of Section 3.6.3 with the Gamma approximation is labelled kPC-Resid. Finally the PC algorithm based on the SNRIC from Section 3.7 is labelled SNR-PC.

4.4.2 Data Simulated from an Artificial Network

The network and relationships between the nodes are described in Figures 4.6a & 4.6b. To generate the data we start by generating values for the parent-less nodes according to their probability distributions provided in Figure 4.6b, we continue by generating values for the nodes whose parents are already generated according to their probability distributions and relationships with parent nodes provided in Figure 4.6b. Since the network contains non-linear relationships and non-Gaussian noise, as expected, the PC algorithm performed worst (area under the ROC curve of 0.76 ± 0.05 averaged over a 100 runs). Close to perfect performance was achieved by dPC (0.96 ± 0.04) and kPC-Clust (0.98 ± 0.05), though SNR-PC (0.92 ± 0.03) and kPC-Resid (0.89 ± 0.05) showed very similar results. Figure 4.7 shows the mean ROC curves (solid lines) and one standard deviation spreads (dashed lines) for repeatedly simulated data. Table 4.7 shows how many times algorithms outperformed each other out of 100 runs. All independence criteria based algorithms outperformed the traditional PC $98 - 100$ times out of 100. Note that the number of times the algorithm i outperformed the algorithm j and vice versa does not have to add up to 100 as on some datasets they might perform equally well, i.e. get the same area under the ROC curve. This happens quite often due to the small number of edges.

4.4.3 Data Simulated by Re-sampling

Next we compare the performance of the algorithms on samples obtained by fitting a plausible network to the single-cell data in Sachs et al. (2005) and re-sampling residuals.



(a) Graph of the toy simulated example on 9 nodes

$$
\begin{aligned}
X_1 &\sim U[0, 10] \\
X_2 &\sim U[0, 3] \\
X_3 &\sim \sin(X_1) + X_2 + 0.6U[0, 1] \\
X_4 &\sim N[0, 1] \\
X_5 &\sim X_3 + X_4 + 2U[0, 1] \\
X_6 &\sim N[0, 1] \\
X_7 &\sim N[0, 1] \\
X_8 &\sim X_6 + X_7^3 + N[0, 1] \\
X_9 &\sim X_7^2 + N[0, 1]
\end{aligned}
$$

(b) Relationships between the nodes

Fig. 4.6 Toy simulated example on 9 nodes and 300 observations.

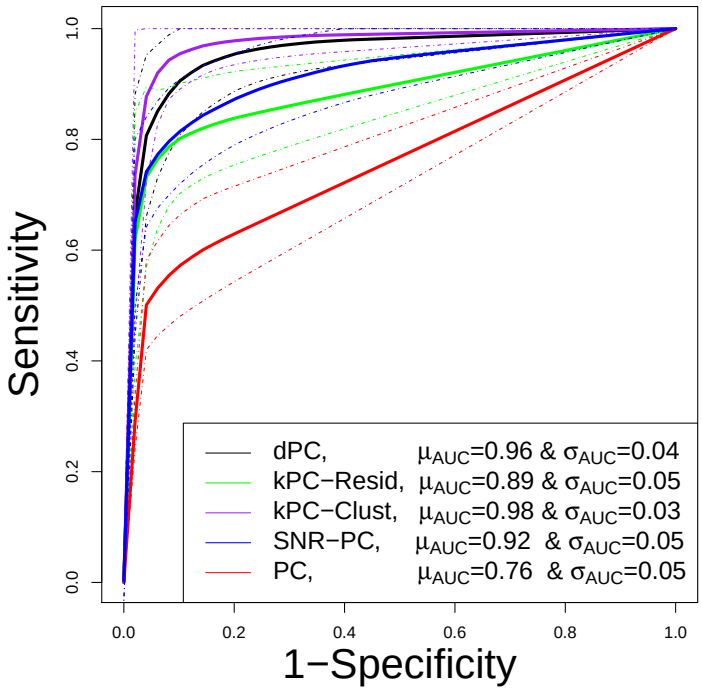


Fig. 4.7 ROC curves to compare kPCs, dPC, SNR-PC and PC algorithms on the toy simulated example. The solid lines are the mean ROC curve and the dotted lines are the mean ROC curve ± one standard deviation.

As we discussed in Section 1.6, we have a reference network for these datasets, given in Figure 1.9b. This allows us to generate a simulated dataset from a real one. There are two benefits of doing so. First of all we still control the structure of the underlying

	dPC	kPC-Resid	kPC-Clust	SNR-PC	PC
dPC	0	84	23	77	100
kPC-Resid	12	0	6	32	98
kPC-Clust	58	88	0	82	100
SNR-PC	18	65	11	0	99
PC	0	2	0	0	0

Table 4.7 Comparison of the PC algorithm versions on a small simulated example. Number in the i^{th} row and the j^{th} column represents how many times algorithm i outperformed the algorithm j.

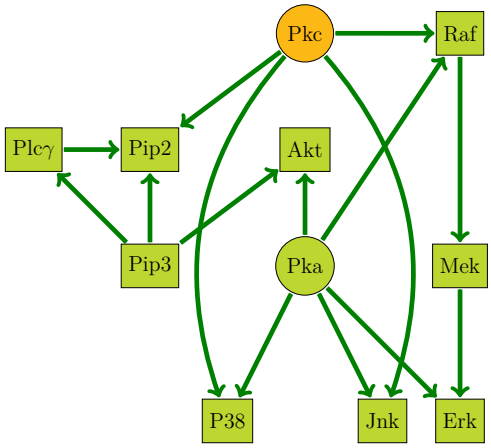
network but obtain more realistic noise distributions. Second instead of one dataset, we can generate many allowing us to run our algorithms multiple times and get confidence interval on any metric we are going to use to evaluate the algorithms.

As outlined in the introduction, the data consist of eight datasets of expression levels of eleven proteins, each dataset obtained after specific experimental interventions. Here we present only the results from the dataset 8, the rest is provided in Appendix D, Figure D.1. Protein PKC was inhibited for dataset 8. Since PKC was externally modified no causal arcs lead into PKC. The network is that of Figure 1.9b with arcs into PKC removed.

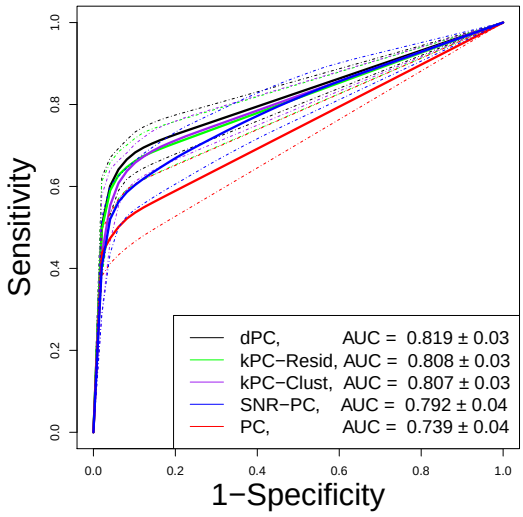
For the simulation we used the causal model of Figure 4.8a. The data generation starts from parent-less nodes (PKC and PKA). These variables are assigned the original values from the samples in the experimental dataset. Next, recursively iterating over nodes whose parents already have assigned values, a generalised additive model is fitted to obtain mean estimates and residuals for the experimental sample values of the focus node when regressing on the values previously assigned to its parents. These residuals are permuted before being added to mean estimates to obtain re-sampled values to assign to the focus node.

This procedure ensures that only the assumed dependencies as captured in the non-linear regressions on parents are maintained, while all other dependencies are removed by permuting residuals. On the other hand, the noise characteristics of the original data are maintained to some degree. In particular, some focus nodes show little functional dependence (see Figure 1.10b, for example PKC $\rightarrow$ RAF shows almost no signal) on their parents, that is, a very low signal to noise ratio. This is a characteristic of experimental data as well. In order to explore the influence of this signal to noise ratio, additional data sets are simulated with residuals scaled down by a factor k before being added to mean estimates, improving on the signal to noise ratio.

Figure A.1 in Appendix A show the dependencies and pairwise scatterplots between variables of the dataset simulated from dataset 8. All the dependencies and scatterplots look almost identical to the ones of the real dataset 8 in Figure 1.10, which gives us some confidence that simulating in this way preserves most of the properties of the original dataset.



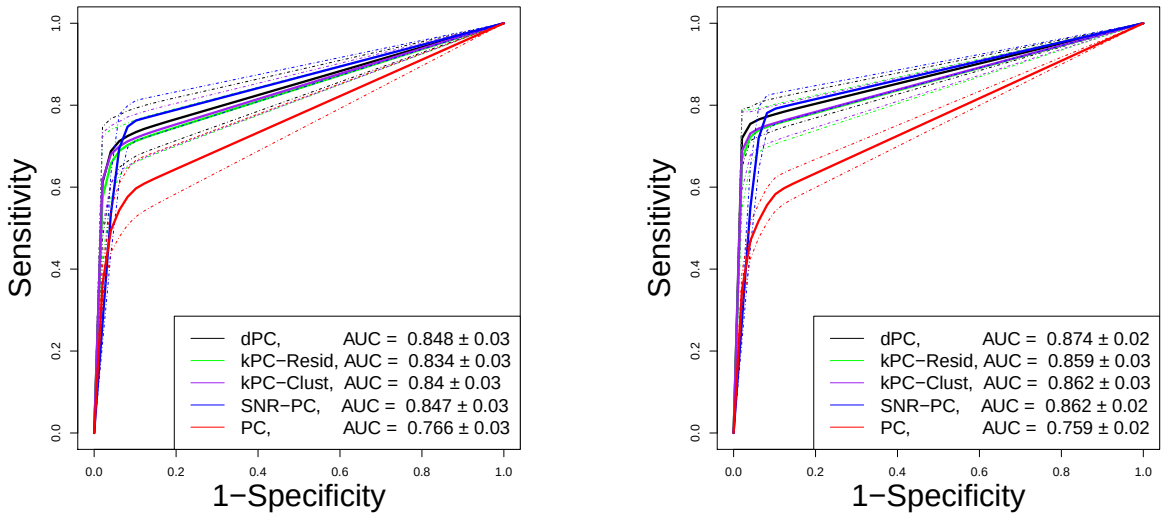
(a) Graph to simulate from the Dataset 8.



(b) ROC curve to compare kPCs, dPC, SNR-PC and PC algorithms on the simulated Dataset 8.

Fig. 4.8 Data simulated with non-reduced noise from dataset 8.

Figure 4.8b shows that all the PC versions based on the general independence criteria significantly outperform the traditional PC algorithm. dPC, kPC-Resid and kPC-Clust result in the mean areas under the ROC curve greater than 0.8, SNR-PC of 0.79 while that of PC is only 0.74. The standard deviation for all algorithms is almost the same, approximately $0.03 - 0.04$. Performance is worse than for the toy example above. This is mainly due to a small signal to noise ratio for many relationships: on visual inspection many relationships in Figure 1.9b are hardly noticeable in the data. This results in the regression step not capturing much signal. On the other hand, real data are likely to show this type of noise characteristics. The effect of varying the scaling factor k for the residual noise is shown in Figure 4.9. Generally, as expected, with lower noise performance improves. Reducing noise 10 times increased the AUC by approximately 0.05, which is significantly more than one standard deviation (≈ 0.03). It would be possible to argue that we are considering only the mean area under the curve but the performance of the PC and kPC algorithms are highly correlated over networks, making the comparisons of mean performance less informative. We are



(a) 3 times reduced noise.

(b) 10 times reduced noise.

Fig. 4.9 ROC curve to compare kPCs, dPC, SNR-PC and PC algorithms on the data simulated with reduced noise from dataset 8.

actually interested in the probability that the kPC algorithm performs better than the PC algorithm. Table 4.8 shows how many times which algorithm outperformed others out of 100 runs on the different datasets simulated from the dataset 8. On the data simulated from dataset 8 the dPC, kPC-Resid, kPC-Clust and SNR-PC outperform the PC algorithm 96, 92, 93 and 89 out of a 100 iterations. Table 4.9 provides the average results over datasets simulated from all 8 datasets (800 simulated datasets in total). kPC algorithms still outperform the original PC algorithm approximately nine times out of ten. Among themselves the kPC versions perform very similarly with dPC having a slight edge over the both kPC versions and SNR-PC.

	dPC	kPC-Resid	kPC-Clust	SNR-PC	PC
dPC	0	60	58	68	96
kPC-Resid	31	0	47	62	92
kPC-Clust	31	45	0	63	93
SNR-PC	25	31	32	0	89
PC	3	8	5	4	0

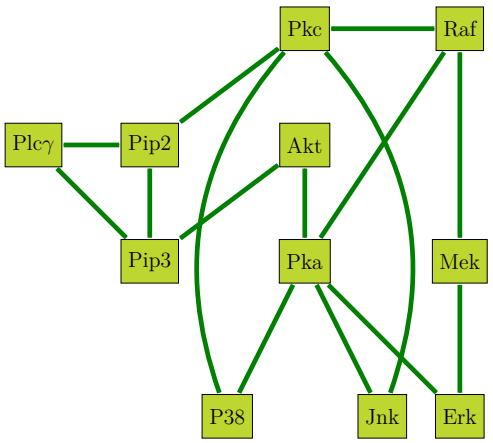
Table 4.8 Comparison of the PC algorithm versions on the data simulated from dataset 8.

	dPC	kPC-Resid	kPC-Clust	SNR-PC	PC
dPC	0.00	56.12	54.38	53.62	90.50
kPC-Resid	32.88	0.00	42.88	46.75	86.62
kPC-Clust	34.88	43.88	0.00	46.75	85.62
SNR-PC	36.50	44.50	44.25	0.00	87.00
PC	5.50	7.00	9.25	8.12	0.00

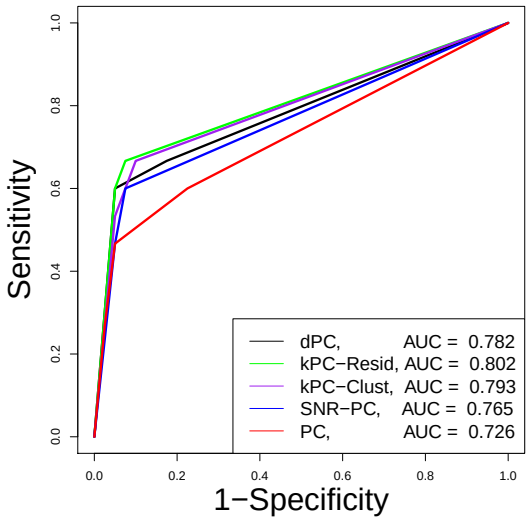
Table 4.9 Comparison of the PC algorithm versions on all 8 simulated datasets.

4.4.4 Original Data

Finally we tested all the algorithms on the original single-cell data from Sachs et al. (2005). Again we only show the results on dataset 8 here, the rest of the results are provided in the Appendix D, Figure D.2. We expect to find the skeleton of the graph in Figure 4.10a derived from Figure 1.9a as in the previous section. In Figure 4.10b we see the ROC curves for five versions of the PC algorithms. Results are very similar to the ones seen for simulated data: dPC, kPCs and SNR-PC outperform PC and are quite similar among themselves. We may conclude that independence criteria based PC versions are a significant improvement on the traditional PC algorithm on the real data as well as the simulated one.



(a) Skeleton of the graph we expect to find from the Dataset 8.

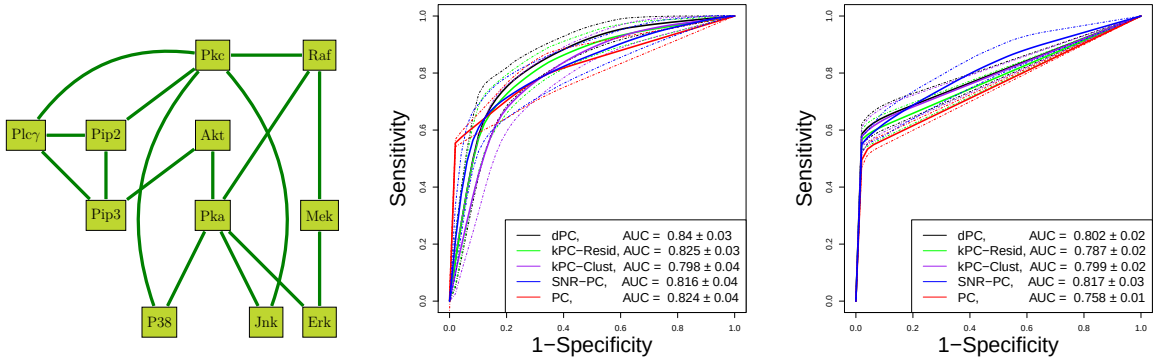


(b) ROC curves to compare kPCs, dPC and PC algorithms on the Dataset 8.

Fig. 4.10 ROC curves for dataset 8.

4.4.5 Combining All Datasets

The eight datasets of Sachs et al. (2005) have slightly different dependence structures due to variation in the external interventions. Combining information from all datasets should improve reconstruction of the underlying graph structure. To test this intuition we combine a consensus graphical structure from networks fitted to each dataset. Two types of consensus networks are obtained. The first takes edges that appear in *at least one* of the individual networks (labelled *union network*). The second calculates the typical average occurrence of edges over all edges in the union network and over all eight networks. Then only those edges of the union network are retained which occur (across all eight networks) more often than this typical average. We label this network *above-average network*. More sophisticated approaches are conceivable, however, here we only wanted to investigate whether there is potential improvement by combining networks at all, and the effect of the choice of an independence criterion on the consensus network. We compare the output of our algorithm to the skeleton illustrated in Figure 4.11a derived from Figure 1.9a and corresponding ROC curves are shown in Figure 4.11b and 4.11c.
Combining networks results in a slight improvement overall compared to Figure 4.10 with a trade-off between sensitivity and specificity shifted between the two types of combinations. The general independence criteria are again superior.



(a) Graph (b) Union network (c) Above-average network

Fig. 4.11 All 8 datasets combined.

4.4.6 Discovering Directions

So far we have looked at performance of algorithms when inferring the skeleton of a DAG with undirected edges only. The PC algorithm adds an edge orienting phase exploiting collider patterns and transitive closure requirements as formalised in the

Meek rules (Meek, 1995) in the collider phase. Exploiting non-linear relationships and non-Gaussian noise additional edges might be oriented. This is achieved by the PC algorithm extended by the generalised transitive phase which incorporates background knowledge emerging from testing directions exploiting non-linearity and non-Gaussian noise.

With an imperfect independence test or data that do not strictly follow modelling assumptions, ambiguities can arise when orienting edges, possibly leading to cycles and doubly oriented edges. There are no general rules how to resolve such ambiguities. In this section we ignore doubly oriented edges as undirected for the purpose of assessing algorithms.

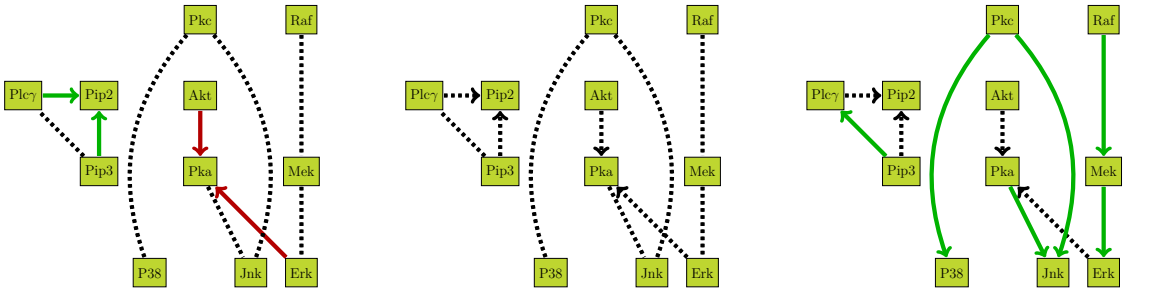
Results comparing algorithms by the number of correct predicted orientations are presented in Table 4.10. Free parameters were fixed as in Table 4.6. The kernel PC using the generalized transitive phase adds many more orientations to those found by the original PC algorithm. To generate Table we used a 100 datasets simulated from the dataset 8 (same as in Section 4.4.3) and took the average over the correctly, wrongly and non-oriented edges. The kPC algorithms orients almost all the edges, with 5.5 (SNR-PC) to 6.5 (kPC) of them being correct and approximately 2 being wrong, while the PC algorithm gives only 3.5 correct, 3 wrong and 1.3 non-oriented edge.

Method	Correctly oriented edges	Wrongly oriented edges	Not oriented edges	Total correct edges
dPC	6.07	1.95	0.14	8.16
kPC-Resid	6.64	1.31	0.11	8.06
kPC-Clust	6.46	1.68	0.09	8.23
SNR-PC	5.43	2.11	0.26	7.80
PC	3.47	3.04	1.34	7.85

Table 4.10 The average number of correctly and wrongly oriented edges as well as not oriented edges for the algorithms.

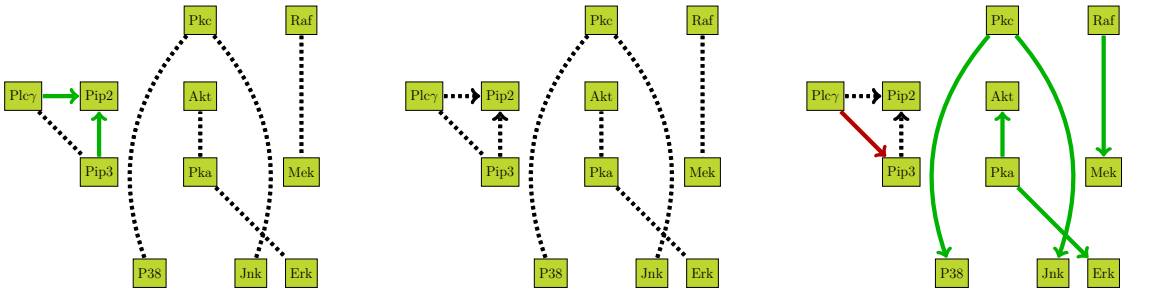
We illustrate some of the results from Table 4.10. Figure 4.12 shows the output graphs of the orientation phases of the algorithm kPC-Resid. In the collider step one v-structure is identified correctly, namely $\text{Plc}_\gamma \rightarrow \text{Pip2} \leftarrow \text{Pip3}$, while another wrongly, namely $\text{Akt}_\gamma \rightarrow \text{Pka} \leftarrow \text{Erk}$. Since there are no more colliders no further edge can be oriented at the transitive step. However, exploiting non-linearity and non-Gaussian noise it is straightforward to orient the rest of the edges in the Generalised transitive phase.

Finally, for comparison with the kPC algorithm, Figure 4.13 shows the output of the dPC algorithm on simulated dataset 8. In contrast to the kPC-Resid algorithm the dPC does not mistake Pka for a collider and therefore is able to correctly orient $\text{Akt}_\gamma \leftarrow \text{Pka} \rightarrow \text{Erk}$ in the generalized transitive step. Though it does orient the edge $\text{Plc}_\gamma \rightarrow \text{Pip3}$ wrongly.



(a) **Skeleton** and **Collider** step (b) **Transitive** step (c) **Generalized transitive** step

Fig. 4.12 Output of the kPC-Resid algorithm on the data simulated from dataset 8. Colour coding: dashed black undirected or doubly directed edges represent correctly identified undirected edges, green directed edges represent correct, while red directed edges represent incorrect orientations. Dashed black oriented edges are from the previous phase.



(a) **Skeleton** and **Collider** step (b) **Transitive** step (c) **Generalized transitive** step

Fig. 4.13 Output of the dPC algorithm on the data simulated from the dataset 8.

4.4.7 Schistosomiasis Dataset

Finally we applied the kPC algorithms to the Schistosomiasis dataset. As we discussed in Section 1.6.3 this dataset has a more complex structure than the single-cell datasets,

i.e. almost every pair of variables is (at least unconditionally) dependent. Therefore we expect a much denser graph. In Figure 4.15 we provide the results of kPC algorithm applied to the Schistosomiasis 11 variable dataset. We do not have a reference network for this network to compare our results with. We present combined outputs of the kPC algorithm at a different cut off $\alpha = 0.1, ..., 0.8$. Recall that low α means that we are less likely to reject the independence hypothesis, i.e. we are more likely to remove an edge. The darkest edges were present in all of the outputs, i.e. we are certain about them, while the lightest edges were present only in the outputs with high α, i.e. we are not very certain about them. Different independence criteria produced similar but not identical results. At the lowest α cut off 12 edges were found by all four methods, another edge by at least three methods and there were 17 edges in total identified by at least one algorithm. All kPC versions found significantly more relationships than the MCMC on the discretized data from Section 2.6.5. This could be explained by the information loss when discretizing the data.

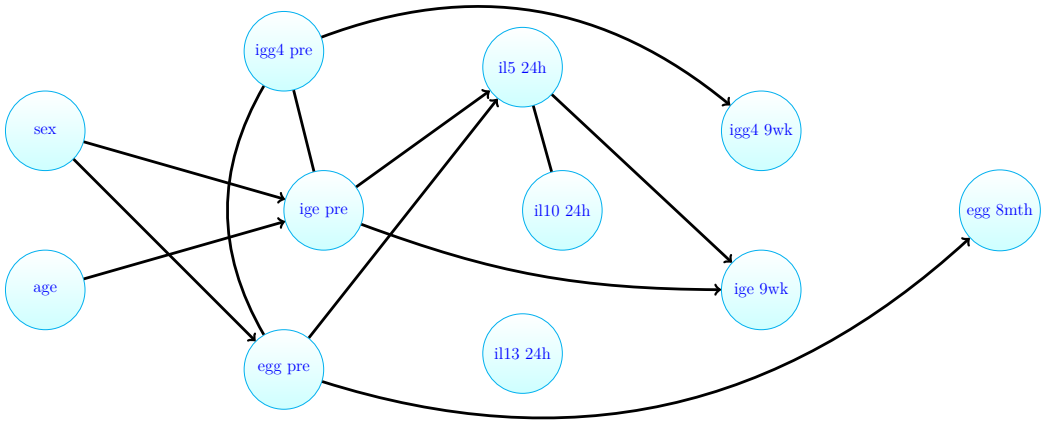
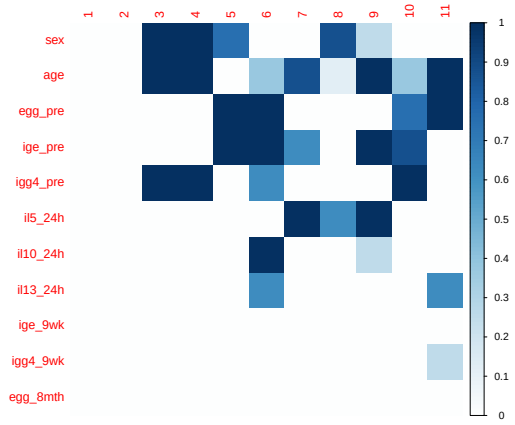


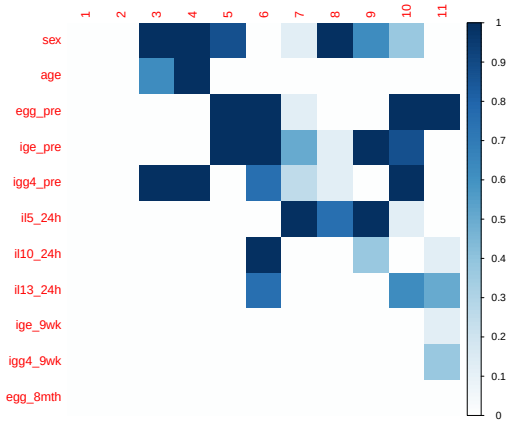
Fig. 4.14 Network for the Schistosomiasis dataset; only the edges present in all four kPC variants with $\alpha = 0.1$ are present.

4.5 Discussion

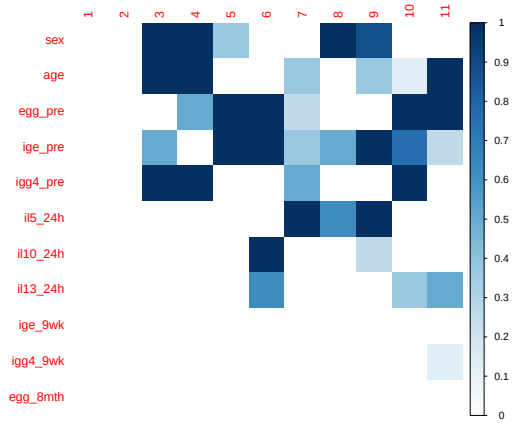
The purpose of this study was to investigate how far probabilistic independence criteria for continuous data that go beyond linear relationships and Gaussian noise can improve the identification of edges and their orientation in a causal graph when applied to experimental data and data simulated in a realistic fashion from experimental data. We analysed two different criteria proposed in the literature, the Hilbert-Schmidt



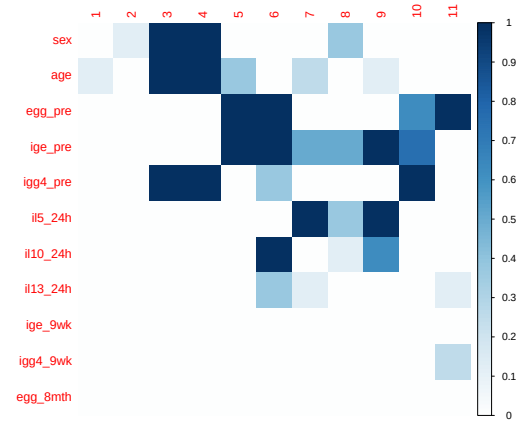
(a) dPC



(b) kPC-Resid



(c) kPC-Clust



(d) SNR-PC

Fig. 4.15 The Schistosomiasis network adjacency matrix found by kPC algorithm using various independence criteria. Matrix M element m_{ij} represents the proportion of outcomes in which the i^{th} node is a parent of the j^{th} node.

Independence Criterion or HSIC, and the Distance Covariance Criterion or dCov, and a Signal to Noise Ratio Independence Criterion or SNRIC in the context of the popular PC algorithm that relies on measures of probabilistic independence for network inference. The distance covariance is a natural extension of the Pearson correlation parameter Székely et al. (2007). To our knowledge this is the first implementation and application of the dCov to causal or network inference. All algorithms discussed in

this study are available as `kpcalg` (Verbyla et al., 2017) package for the R statistical environment (R Core Team, 2014).

Overall, our findings confirm that the performance of general independence criteria is decisively better over that based on linear relationships with Gaussian noise on simulated as well as experimental data in terms of correct undirected edges as well as of correct directions. Secondly, we find only little difference between the performance of the HSIC, dCov and SNRIC in general, with the dCov showing slightly better performance for some datasets.

In order to assess the algorithms in a realistic scenario we applied them to a well-known experimental dataset for which the network is approximately known based on biological knowledge. Of course, this knowledge of the network might be inaccurate and we therefore propose a generic way to simulate data based on experimental data and an approximate or putative network structure that keeps much of the noise characteristics of the original data but reproduces those and only those conditional dependencies required by the network. As we demonstrate in our analysis these simulated datasets form an excellent compromise between retaining much of the non-linear and non-Gaussian characteristics of the original data, but for an exactly known network. As we see in our study one difficulty remains: if some arcs of the assumed network are not supported by the data, for example, if there is little dependency in the data in the first place between two variables which we wish to connect in the network, our method is unable to create such dependency artificially. Nevertheless, as long as the assumed network reflects most of the dependencies in the experimental data, the simulated data are useful for comparative studies between different algorithms as shown in Section 4.4.3.

The PC algorithm requires a test for conditional independence. Independence criteria might, however, only be available in an unconditional form. We propose a simple procedure, based on fitting non-linear regressions, to adapt such criteria to the conditional independence case. Since there is a conditional version of the HSIC available we had an opportunity to compare the conditional HSIC (in algorithm kPC-Clust) with our adaptation of the unconditional HSIC (in algorithm kPC-Resid). As can be seen throughout the study, the adapted kPC-Resid version, particularly on experimental and realistically simulated data, is performing comparably to the conditional version with the conditional HSIC having a slight advantage. It is worth noting that kPC-Clust involves calculating the empirical estimate of the conditional HSIC (3.6.3) which is computationally significantly more expensive than the unconditional HSIC (3.4.12),

therefore in practice kPC-Clust can be up to $5 - 10$ times slower than kPC-Resid or dPC.
The empirical estimation of HSIC depends on parameters such as the kernel width λ and the regularisation parameter ϵ. Here we used simulations to find sensible ranges for these parameters. It should be noted that the HSIC is not scale invariant and therefore the parameter ranges we propose are only suitable for normalized data. Another advantage of the dCov is that it is less affected by parameter choices, essentially only the power parameter ξ which can be safely set to a value between 0.1 and 1 without affecting the results too much. dCov is scale invariant, therefore values for ξ works for any data.
The PC algorithm is very restrictive in its assumptions on the dependency structure. For example, cycles or unobserved variables are excluded. It would be interesting to see whether inference techniques allowing such more complex assumptions benefit from general independence criteria in the same way the PC algorithm does.
The PC algorithm is firmly based in a frequentist statistical framework. Bayesian inference is often strongly dependent on specific noise models through the likelihood function. It needs to be explored how to incorporate independence criteria in a Bayesian framework, possibly through a form of loss likelihood (Bissiri et al., 2016). We will explore this path in the next chapter. A different approach would be to test the effect of different kernels in the HSIC, for example linear, polynomial, hyperbolic tangent, Laplacian or Bessel. They are conveniently implemented in `kernlab` (Karatzoglou and Smola, 2003) package for the R statistical environment (R Core Team, 2014).

4.5.1 Model Limitations

In this chapter we have shown that the general independence criteria based PC algorithm is an efficient tool in inferring networks structure and outperforms the conventional PC algorithm most of the time. The main limitations of this algorithm comes from the framework of the PC algorithm.

1. First of all ICPC is a greedy algorithm and as every greedy algorithm it is not guaranteed to find the global maximum (the optimal network structure) just the local maximum.

2. Algorithm has hyper-parameters, such as maximum number of parents and the independence cut-off, which have to be chosen prior to running the algorithm and the output might be sensitive to these choices.

3. ICPC is susceptible to the existence of latent variables.

4. ICPC cannot deal with cycles.

5. Finally it is important to point out that the current implementation is using regression based unconditional independence testing for the conditional independence tests (Section 3.6.3). We have shown that this works well when data follows some functional relationships, but in principle this is not strictly correct.

4.5.2 Future work

Based on the model limitations discussed in Section 4.5.1 we propose the following paths of possible future extensions. First four limitations are the same as for the original PC algorithm and it is not in the scope of this work to deal with them. We only propose possible extensions with respect to the last limitation.

1. The first step would be to implement the general independence criteria based tests in `C++` or other compiled programming language. This would greatly increase the computational time of the algorithm.

2. Another avenue for the future work is more theoretical rather than programming based. Currently we are only aware of the Cluster-HSIC conditional independence test which is significantly slower than the residual counterparts. ICPC method would strongly benefit if we could develop a fast full conditional independence test and use it instead of either the residuals based independence tests or the Cluster-HSIC.

Chapter 5

MCMC Sampling Using Loss Function

5.1 Introduction/Motivation

So far we have discussed approaches to infer a network structure, when the true underlying graph structure is a directed acyclic graph. In real life biological datasets this is a very strong assumption, which fails more often than not. Occasionally we may ignore the existence of the cyclic structures and still hope to get a meaningful result, but this is more of an exception, rather than a rule. In this chapter we present a model capable of dealing with datasets generated from the networks that have cycles. As we have discussed in section 1.1, networks with feedback loops inference is a particularly complex problem.

In Chapter 2 we introduced an MCMC sampler for network structures, in Chapter 3 we introduced a quantitative way of estimating dependence in the sample: general independence criteria, in Chapter 4 we showed the potential of using these independence criteria in the network inference. Finally in this chapter we are bringing all the pieces together to build an algorithm that can infer networks with non-linear relationships, feedback loops and non-Gaussian noise.

5.2 Model

We start by describing the model that we assume is generating the data. We would like our model to have the following properties:

- to be as flexible as possible, we would like our model to be able to capture all possible relationships between the data. Therefore we want to allow flexible non-linear relationships, possible cyclic structures and any noise.
- to be parametric or semi-parametric, we would like to be able to use our model for future predictions or at least to be able to evaluate it on a left out test subset of the data.
- to be able to estimate the parameters of our model.

Combining these desired properties, we decided to use an additive noise model.

5.2.1 Structural Equation Model

First we introduce a Structural Equation Model (SEM), for more detail see Murphy (2012). SEM is a special kind of directed mixed graph where all relationships are linear Gaussian and cycles are allowed. SEMs are widely used, especially in economics and social science. Usually a directed edge is interpreted as a causality and a directed cycle is interpreted as a feedback loop (for example Pearl (2000)).
Suppose we have n variables $X = (X_1, ..., X_n)$, then SEM can be defined as a series of full conditionals:

$$X_i = \mu_i + \sum_{j=1}^{n} w_{ij} X_j + \epsilon_i, \text{ where } \epsilon \sim \mathcal{N}(0, \Sigma)$$

Now if $I - W$ is invertible we can rewrite the above as:

$$X = (I - W)^{-1}(\mu + \epsilon)$$

And so the joint probability distribution is given by $X \sim \mathcal{N}(\hat{\mu}, \hat{\Sigma})$, where $\hat{\mu} = (I - W)^{-1}\mu$ and $\hat{\Sigma} = (I - W)^{-1}\Sigma(I - W)^{-T}$. This means that x is distributed as a joint Gaussian and there is nothing inherently causal in it.

Example

Here is a small example of a structural equation model on 4 nodes, also shown in Figure 5.1.

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 0 & 0.8 & 0 & 0 \\ 0.7 & 0 & 0 & 0 \\ 0.8 & 0 & 0 & 0 \\ 0 & -0.5 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} + \begin{pmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ \epsilon_4 \end{pmatrix}$$

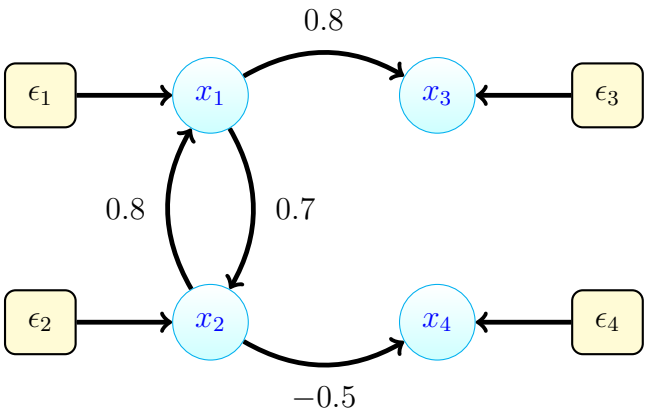


Fig. 5.1 Example of a Structural Equation Model.

5.2.2 Additive Noise Model

The Structural Equation Model is not very flexible, it allows only linear relationships and Gaussian noise, both are strong assumptions that are very likely to fail in a real life example. The additive noise model (Hoyer et al., 2009) we are using can be thought of as an extension of the SEM. We relax the assumption of the linear relationships and Gaussian noise, i.e. now we allow any non-linear relationship between the variables and any noise distribution, while still allowing cyclic structures in the model, just like in a SEM.

$$X_i = \sum_{j=1}^{n} f(X_j, \theta_{ij}) + \epsilon_i$$

Here we assume all ϵ_i to be independent of each other. Note that a SEM is a special case of an additive noise model, that is if we take the relationship f to be linear, i.e. $f(x, \theta) = \theta x$ and the noise to be Gaussian, i.e. $\epsilon \sim \mathcal{N}(0, \Sigma)$ this becomes exactly a SEM.

Example

We present a simple example showing the importance of correctly identifying a cycle in the network. Suppose data is generated by a simple cyclic model from Equation 5.2.1.

$$\begin{aligned} X &\sim \alpha Y + \epsilon_X;\ \epsilon_X \sim N(0,1) \\ Y &\sim \beta X + \epsilon_Y;\ \epsilon_Y \sim N(0,1) \end{aligned} \tag{5.2.1}$$

We are interested in the effect of X on Y. Suppose we overlook the cyclic relationship between X and Y and try to fit a non-cyclic model $Y \sim \hat{\beta}X + \epsilon$. Using a least square regression to find $\hat{\beta}$, we would have to minimize

$$\min_{\hat{\beta}} (Y - \hat{\beta}X)^2$$

Using a substitution we can rewrite Equation 5.2.1 as

$$\begin{aligned} X &= \alpha\beta X + \alpha\epsilon_Y + \epsilon_X = \frac{1}{1-\alpha\beta}\left(\epsilon_X + \alpha\epsilon_Y\right) \\ Y &= \alpha\beta Y + \beta\epsilon_X + \epsilon_Y = \frac{1}{1-\alpha\beta}\left(\beta\epsilon_X + \epsilon_Y\right) \end{aligned}$$

To find $\hat{\beta}$ we consider the expectation

$$\begin{aligned} \mathrm{E}(Y - \hat{\beta}X)^2 &= \mathrm{E}\left[\frac{1}{(1-\alpha\beta)^2}\left((1-\alpha\hat{\beta})\epsilon_Y + (\beta - \hat{\beta})\epsilon_X\right)^2\right] \\ &= \frac{1}{(1-\alpha\beta)^2}\left((1-\alpha\hat{\beta})^2\mathrm{E}[\epsilon_Y^2] + 2(1-\alpha\hat{\beta})(\beta-\hat{\beta})\mathrm{E}[\epsilon_X\epsilon_Y] + (\beta-\hat{\beta})^2\mathrm{E}[\epsilon_X^2]\right) \\ &= \frac{1}{(1-\alpha\beta)^2}\left((1-\alpha\hat{\beta})^2 + (\beta-\hat{\beta})^2\right) \end{aligned}$$

After differentiating with respect to $\hat{\beta}$ and equating to zero we get $\hat{\beta} = \frac{\beta+\alpha}{1+\alpha^2}$. $\hat{\beta} = \beta$ if and only if $\alpha = 0$, i.e. the relationship is truly not cyclic. Though suppose the true parameters are: $\alpha = -0.6$ and $\beta = 0.4$, then $\hat{\beta} \approx -0.15$. Not only the magnitude of the estimate is wrong but the sign as well. If we are interested in parameters of the relationships (for example we are interested in predictive power), then missing a cycle can lead to very wrong conclusions.

5.3 Theory

Markov Chain Monte Carlo (MCMC) methods are very popular (Tasaki et al., 2015). They overcome issues such as the necessity to calculate the probability for every sample point. For example the often used Metropolis-Hastings algorithm requires only the ratio of probabilities at two points, so we avoid calculating the normalising constant for the probability distribution. For a detailed discussion of using MCMC sampler for structure search see Section 2.2. There are two possible issues that may arise. Firstly how do we sample from models that have different numbers of parameters, for example in the network inference the number of parameters depend on the network structure (the number of edges in the network). The other problem arises if we do not have a probability distribution to sample from. For example if we are using an additive noise model and we do not assume a probability distribution for the noise terms, then we are unable to define a probability distribution for our model. It could also be the case that just calculating the ratio of probabilities is already computationally infeasible.
In the following section we discuss possible approaches to work around these problems. We first introduce a reversible jump MCMC in Section 5.4 and then a possibility to sample from a loss function instead of the likelihood in Section 1.4.

5.4 Reversible Jump MCMC

Using an MCMC sampler in order to find a network involves sampling the network structure as well as sampling the parameters for each structure. This means that when the MCMC chain jumps to a different structure it effectively changes the dimensionality of the space it is sampling from. This raises a question how to form the proposal distribution in order that the MCMC chain could sample both correctly and efficiently. The problem of the correct sampling distribution is discussed in Green (1995) and how to form an efficient jump proposal which would be accepted with a high probability is discussed in Brooks et al. (2003).

5.4.1 Jump to a Higher Dimensional Space

Suppose we want to propose a jump from the state in a model M_i with the parameters θ_i (which is from the n_i dimensional space Θ_i) to the state in a higher dimensional model M_j with the parameters θ_j (which is from the n_j dimensional space Θ_j), i.e. $n_j > n_i$. In our network inference case this happens when we want to propose to add an edge. Green (1995) proposes to generate an $n_j - n_i$ dimensional random vector v

from some proposal density $\phi_{n_j-n_i}(v) = \prod_{k=1}^{n_j-n_i} \phi(v_k)$. Then to propose the next state $\theta_j = f_{i,j}(\theta_i, v)$, using some function $f_{i,j} : \Theta_i \times \mathbb{R}^{n_j-n_i} \to \Theta_j$. This move is accepted with probability

$$P\left\{(M_i, \theta_i), (M_j, \theta_j)\right\} = \min\left(1, A_{i,j}(\theta_i, \theta_j)\right) \tag{5.4.1}$$

where

$$A_{i,j}(\theta_i, \theta_j) = \frac{\pi(M_j, \theta_j)}{\pi(M_i, \theta_i)} \frac{r_{ji}(M_j, \theta_j, M_i)}{r_{ij}(M_i, \theta_i, M_j)} \frac{1}{\phi_{n_j-n_i}(v)} \left| \frac{\partial f_{i,j}(\theta_i, v)}{\partial(\theta_i, v)} \right| \tag{5.4.2}$$

$r_{ij}(M_i, \theta_i, M_j)$ is the probability to propose a jump from the model M_i with parameters θ_i to the model M_j, for more details see Section 2.2. We suppose that all the models $M_1, ..., M_i, ...$ have the priors $p(M_1), ..., p(M_i), ...$ respectively. We also suppose that the model M_i has the parameters θ_i with the priors $p_i(\theta_i), \theta_i \in \Theta_i$. Finally given the model M_i and its parameters θ_i the likelihood of the data x is given by $L_i\left(x|\theta_i\right)$. So the posterior density is given by

$$\pi\left(M_i, \theta_i\right) \propto L_i\left(x|\theta_i\right) p_i(\theta_i) p(M_i)$$

Example: Simple nested models

Suppose we have simple nested models. That is $n_i = k$, $n_j = k + 1$ and we are using the identity function f, i.e. $f_{k,k+1}(\theta_k, v) = (\theta_k, v)$. Also suppose we draw our proposal v from $N(0, \sigma^2)$. Then (5.4.2) simplifies to

$$A_{k,k+1}(\theta_k, \theta_{k+1}) = \frac{L_{k+1}(x \mid \theta_k, v)}{L_k(x \mid \theta_k)} \frac{p_{k+1}\left((\theta_k, v)\right)}{p_k(\theta_k)} \frac{p(M_{k+1})}{p(M_k)} \frac{r_{k+1,k}(M_{k+1}, M_k)}{r_{k,k+1}(M_k, M_{k+1})} \frac{1}{\phi(v)} \tag{5.4.3}$$

where

$$\phi(v) = (2\pi\sigma^2)^{-1/2} \exp(-v^2/2\sigma^2)$$

and the proposal $r_{k,k+1}$ is independent of the parameters θ_k.

Adding an Edge

As stated before, in the case of the network inference a jump to a higher dimensional space is necessary when adding an edge. Suppose we propose to add an edge $\{x \to y\}$, i.e. to move from the graph G_i to the graph $G_j = G_i \cup \{x \to y\}$. In this case we

would also need to propose the parameter(s) θ_{xy} for the edge $\{x \to y\}$. Here we can use the proposal function f as an identity function, i.e. the proposed state is $\theta_j = f_{i,j}(\theta_i, v) = (\theta_i, v)$. In this case the acceptance probability $A_{i,j}$ from Equation 5.4.2 simplifies to:

$$A_{i,j}(\theta_i, \theta_j) = \frac{\pi(M_j, \theta_j)}{\pi(M_i, \theta_i)} \frac{r_{ji}}{r_{ij}} \frac{1}{\phi_n(v)} \tag{5.4.4}$$

here ϕ_n is the proposal distribution for the parameters of the new edge and n is the number of parameters per edge (one in the case of linear model and the number of the linear elements in the piece-wise linear model). We also dropped the dependence of the structure change proposal r_{ij} on the parameters θ_i (equivalently for the r_{ji}) as this depends only on the structure of the graph G_i and not on its parameters θ_i

5.4.2 Centring Proposals

In order to be able to deal with cases more complicated than the simple nested model, Brooks et al. (2003) suggests to introduce a set of centring functions. A centring function $c_{ij} : \Theta_i \to \Theta_j$ can be defined as

$$c(\theta_i) = f\left(\theta_i, b(\theta_i)\right)$$

where $b(\theta_i) = b_{ij}(\theta_i)$ is some real-valued function, often taken to be identically zero. This defines a special point $b(\theta_i)$ for the proposal vector v which is mapped to a special point $c \in \Theta_j$ in the higher dimensional space Θ_j. There are various ways to choose the centring functions c (which essentially is defined by the choice of the function b). We discuss two options proposed by Brooks et al. (2003):

1. Weak non-identifiability
2. Conditional maximization

5.4.3 Weak Non-Identifiability Approach

Suppose we want to make a jump from a model M_i with the parameters $\theta_i \in \Theta_i$ to a higher dimensional model M_j with the parameters $c(\theta_i) \in \Theta_j$. The weak non-identifiability approach relies on finding the jump proposal $c(\theta_i)$ such that the probabilistic models (in terms of likelihood) defined by $\theta_i \in \Theta_i$ and $c(\theta_i) \in \Theta_j$ are identical. For example in the SEM case, models θ_i and $(\theta_i, 0)$ are identical in terms of the likelihood.

5.4.4 The Conditional Maximization Approach

An alternative to the weak non-identifiability approach is a so called conditional maximization. Our overall goal is to choose a proposal which is likely to be accepted. Therefore when considering where to jump to in the higher dimensional space, an obvious choice for the proposal is the posterior modes in the higher dimensional model. The conditional maximization approach is to maximize the posterior distribution $\pi\left(M_j, h(\theta_i, u)\right)$ with respect to u to obtain the maximizing value $\hat{u}$, say, i.e.

$$\hat{u} = \underset{u}{\operatorname{argmax}}\ \pi\left(M_j, h(\theta_i, u)\right)$$

Then our centring point is chosen so that $c(\theta_i) = h(\theta_i, \hat{u})$. Thus, we are essentially conditioning on the current state θ_i and centring at the posterior conditional mode.

5.4.5 Zeroth-Order Method

Now we shall introduce the simplest method for choosing the proposal distribution. Suppose that we are currently in a state θ_i and we wish to sample for v which can be used to generate a state in the new model, θ_j (recall that $\theta_j = f(\theta_i, v)$). We choose the distribution of v so that, for the jump between θ_i and its image in θ_j under the centring function $c(\theta_i)$, the acceptance ratio given in equation 5.4.2 is identically equal to 1, i.e.

$$A_{ij}\left(\theta_i, c(\theta_i)\right) = 1 \tag{5.4.5}$$

Usually we are choosing the distribution of v from a family of canonical distributions with a low number of parameters (for example a normal $\mathcal{N}(\mu, \sigma^2)$ or uniform $U[-\alpha, \alpha]$). Another option suggested by (Brooks et al., 2003) is to use the higher order methods. For example a first order method would require to simultaneously solve both:

$$\begin{aligned} A_{ij}\left(\theta_i, c(\theta_i)\right) &= 1 \\ \nabla_v A_{ij}\left(\theta_i, c(\theta_i)\right) &= 0 \end{aligned}$$

The higher order methods give more accurate proposals (the probability to accept the jump to a higher dimensional model is higher) but they are computationally more expensive.

5.4.6 Example 1: Weak Non-Identifiability

We present a simple example to make the concept of the zeroth-order centring more clear. Lets consider a SEM (as described in Section 5.2.1) on n nodes, i.e. we assume that our data x is generated by the following model:

$$x_i = \mu_i + \sum_{j=1}^{n} w_{ij} x_j + \epsilon_i, \text{ where } \epsilon \sim \mathcal{N}(0, \Sigma)$$

In this case the possible models M are all possible graph structures on n nodes. There are $n(n-1)$ possible edges, and therefore there are $2^{n(n-1)}$ of possible graph structures. Lets assume a uniform prior on these models $p(M_i) = 2^{-n(n-1)}, \forall i = 1, ..., 2^{n(n-1)}$.
An n_i dimensional model M_i has parameters θ_i (in this case these are the weights w_{ij}'s of the non-zero edges of the graph). Lets assume the prior for θ_i to be

$$p_i(\theta) = (2\pi\sigma_a^2)^{-n_i/2} \prod_{k=1}^{n_i} \exp(-\theta_k^2/2\sigma_a^2)$$

Now suppose we want to propose to add an edge to the current model M_i which already has k edges. Then $n_i = k$ and $n_j = k+1$. Suppose we do that by drawing a random variable v from the normal distribution, i.e. $v \sim q = \mathcal{N}(0, \sigma^2)$. We are using the identity function for the next state generation, i.e. $f(\theta_i, v) = (\theta_i, v)$, and therefore $\left|\frac{\partial f_{i,j}(\theta_i,v)}{\partial(\theta_i,v)}\right| = 1$. Now the likelihoods $L_k(X \mid \theta_k)$ and $L_{k+1}(X \mid (\theta_k, 0))$ are equal, so we expect to accept this jump with probability 1.
Equation (5.4.2) simplifies to

$$A_{k,k+1}(\theta_k, (\theta_k, 0)) = \frac{L_{k+1}(x \mid \theta_k, 0)}{L_k(x \mid \theta_k)} \frac{p_{k+1}(\theta_k, 0)}{p_k(\theta_k)} \frac{p(M_{k+1})}{p(M_k)} \frac{r_{k+1,k}}{r_{k,k+1}} \frac{1}{q(0)} \tag{5.4.6}$$

Note that $L_{k+1}(x \mid \theta_k, 0) = L_k(x \mid \theta_k)$ and $p(M_{k+1}) = p(M_k)$, then setting the acceptance ratio to 1, Equation 5.4.6 simplifies to

$$A_{k,k+1}(\theta_k, (\theta_k, 0)) = \frac{(2\pi\sigma_a^2)^{-1/2}}{1} \frac{r_{k+1,k}}{r_{k,k+1}} \frac{1}{(2\pi\sigma^2)^{-1/2}} = 1 \tag{5.4.7}$$

We can solve the above 5.4.7 for σ to obtain

$$\sigma^2 = \sigma_a^2 \left(\frac{r_{k,k+1}}{r_{k+1,k}}\right)^2 \tag{5.4.8}$$

So it follows that in the above SEM model, if we attempt to make a jump to a higher dimensional model by adding an edge and we sample the parameter for that edge

from the distribution $\mathcal{N}\left(0, \sigma_a^2 \left(\frac{r_{k,k+1}}{r_{k+1,k}}\right)^2\right)$, then the edge with a parameter 0 would be accepted with the probability 1.

5.4.7 Example 2: Conditional Maximization

Lets consider the same set up as in the previous example. Let $p(M_i)$, $p_i(\theta)$ and u be same as before. But this time lets use $v \sim q = \mathcal{N}(\mu, \sigma^2)$, for some fixed μ (suppose we are sampling the parameter for an edge $X \to Y$, then we could use $\mu = \frac{\text{Cov}(X,Y)}{\text{Var}(X)}$ in order to maximize the acceptance of the jump).

$$\begin{aligned} A_{k,k+1}(\theta_k, (\theta_k, \mu)) &= \frac{L_{k+1}(x \mid \theta_k, \mu)}{L_k(x \mid \theta_k)} \frac{p_{k+1}(\theta_k, \mu)}{p_k(\theta_k)} \frac{p(M_{k+1})}{p(M_k)} \frac{r_{k+1,k}}{r_{k,k+1}} \frac{1}{q(\mu)} \\ &= \frac{L_{k+1}(x \mid \theta_k, \mu)}{L_k(x \mid \theta_k)} \frac{(2\pi\sigma_a^2)^{-1/2} \exp(-\mu^2/2\sigma_a^2)}{1} \frac{r_{k+1,k}}{r_{k,k+1}} \frac{1}{(2\pi\sigma^2)^{-1/2}} \end{aligned} \tag{5.4.9}$$

We can solve this for σ to obtain

$$\sigma^2 = \sigma_a^2 \exp(\mu^2/\sigma_a^2) \left(\frac{L_k(\theta_k)}{L_{k+1}((\theta_k, \mu))}\right)^2 \left(\frac{r_{k,k+1}}{r_{k+1,k}}\right)^2$$

This has a greater computational cost, because we need to calculate the ratio of the likelihoods. This is a viable choice as we are proposing a direct jump to the mode of distribution.

5.4.8 Jump to a Lower Dimensional Space or Removing an Edge

So far we have discussed the reversible MCMC jump from a lower dimensional space to a higher dimensional space. In the context of the network inference this would represent adding an edge. Now we discuss the acceptance probability of removing an edge, i.e. the proposal of going from a higher dimensional space to a lower dimensional space.

Suppose we are currently in a graph G_j and we propose to remove an edge $\{x \to y\}$ with parameters θ_{xy}, i.e. we propose a jump from the graph G_j with parameters θ_j to the graph $G_i = G_j \backslash \{x \to y\}$ with parameters θ_i. We can rewrite this as $G_j = G_i \cup \{x \to y\}$ and $\theta_j = (\theta_i, \theta_{xy})$ in order to keep it consistent with the jump to the higher dimensional space from Section 5.4.1. We accept this jump with the probability

$$P\left\{(G_j,\theta_j),(G_i,\theta_i)\right\} = \min\left(1, A_{j,i}(\theta_j,\theta_i)\right) = \min\left(1, A_{i,j}^{-1}(\theta_i,\theta_j)\right)$$

where

$$A_{i,j}(\theta_i,\theta_j) = \frac{\pi(G_j,\theta_j)}{\pi(G_i,\theta_i)}\frac{r_{ji}}{r_{ij}}\frac{1}{\phi_n(\theta_{xy})}$$

just like in Section 5.4.1, Equation 5.4.4, where θ_{xy} are the parameters for the edge $\{x \to y\}$ that we propose to remove and the ϕ_n is the probability distribution that we would use for the proposal of the parameters for the edge $\{x \to y\}$ if we would propose to make a jump from G_i to $G_j = G_i \cup \{x \to y\}$.

5.4.9 Inverting an Edge

In the network inference MCMC sampling a useful jump proposal is to invert an edge. Suppose we want to invert an edge $x \to y$, i.e. we are proposing a jump from a graph $G_i = G_k \cup \{x \to y\}$ with parameters (θ, θ_{xy}) to $G_j = G_k \cup \{y \to x\}$ with parameters (θ, θ_{yx}). We can think about this as first removing an edge $x \to y$ and then adding an edge $y \to x$., i.e. $G_i \to G_k \to G_j$.
Lets suppose that we are using the weak non-identifiability centring. In this case we would be proposing the jump from G_i to G_k with the acceptance probability:

$$A_{i,k}\left((\theta,\theta_{xy}),\theta\right) = \frac{\pi(G_k,\theta)}{\pi\left(G_i,(\theta,\theta_{xy})\right)}\frac{r_{ki}}{r_{ik}}\frac{\phi_n(\theta_{xy})}{1}$$

and the jump from G_k to G_j with the acceptance probability:

$$A_{k,j}\left((\theta,\theta_{xy}),\theta\right) = \frac{\pi\left(G_i,(\theta,v)\right)}{\pi(G_k,\theta)}\frac{r_{jk}}{r_{kj}}\frac{1}{\phi'_n(v)}$$

Now we can combine the above to get the acceptance probability for the jump from G_i with parameters (θ, θ_{xy}) to G_j with parameters (θ, v) as

$$A_{i,j}\left((\theta,\theta_{xy}),(\theta,v)\right) = \frac{\pi\left(G_i,(\theta,v)\right)}{\pi\left(G_i,(\theta,\theta_{xy})\right)}\frac{r_{ji}}{r_{ij}}\frac{\phi_n(\theta_{xy})}{\phi'_n(v)}$$

in this case v is the proposed parameters for the (inverted) edge $y \to x$. Note that the parameter proposal distributions ϕ_n and ϕ'_n are not the same in general. In order to preserve the weak non-identifiability centring we have:
In the $n = 1$ case:

$$\phi_n \sim \mathcal{N}\left(0, \sigma_a^2 \left(\frac{r_{ki}}{r_{ik}}\right)^2\right)$$
$$\phi'_n \sim \mathcal{N}\left(0, \sigma_a^2 \left(\frac{r_{jk}}{r_{kj}}\right)^2\right)$$

Otherwise

$$\phi_n \sim \mathcal{N}\Big(0, \Sigma\left(\sigma_a, r_{ki}, r_{ik}\right)\Big)$$
$$\phi'_n \sim \mathcal{N}\Big(0, \Sigma\left(\sigma_a, r_{jk}, r_{kj}\right)\Big)$$

5.5 Sampling from the Loss Function

Suppose we have observations x from a (semi) parametric model

$$X \sim f(\theta, \epsilon)$$

where X is the random variable that we can observe, f is some function that we know, θ are some parameters that we want to infer and ϵ is some random noise that we cannot observe.
We are interested in the estimation of the unknown parameters θ. We want to use a Bayesian set up and we have a prior $\pi(\theta)$ for θ. The usual approach would be to use the likelihood

$$L(\theta \mid x) = f_\theta(x)$$

This would allow us to obtain the posterior probability distribution $p(\theta \mid x)$ (for example by using the MCMC Metropolis Hastings sampling) from the

$$p(\theta \mid x) \propto L(\theta \mid x)\pi(\theta)$$

But what happens if we do not have $f_\theta(x)$ and therefore we do not have the likelihood $L(\theta \mid x)$? It is possible that even though we do not have a likelihood, we still may have some loss function $l(x, \theta)$ (for full detail see Section 1.4). For example a loss function based on some independence criterion, as described in Section 5.6.2. Now we explore how we can obtain a probability distribution over the parameters θ given observed data x using a loss function $l(x, \theta)$. This approach is taken from Bissiri et al. (2016). A natural first step is to try to minimise the expected loss $R(x, p)$, where p is the probability density function of the parameters θ. That is we are interested in finding a

distribution p over the parameters θ, which minimises the expected loss, that is we want to solve:

$$\begin{aligned}\hat{p}(\theta) &= \operatorname*{argmin}_p R(x,p) \\ &= \operatorname*{argmin}_p \int l(x,\theta)p(\theta)d\theta\end{aligned}$$

This is minimised by the Dirac delta function δ_α where α is the value of θ that minimises the loss function $l(x,\theta)$, i.e.

$$\begin{aligned}&\hat{p} \sim \delta_\alpha, \\ &\alpha = \operatorname*{argmin}_\theta l(x,\theta)\end{aligned}$$

This is not a particularly interesting result from a Bayesian point of view as this does not provide us with an interesting distribution to sample from. The Dirac delta function has all its probability mass in one point. This suggests to upgrade our approach by introducing the entropy: try finding a distribution p that simultaneously minimises the expected loss and maximises the entropy:

$$\begin{aligned}\hat{p}(\theta) &= \operatorname*{argmin}_p R(x,p) + H(\theta) \\ &= \operatorname*{argmin}_p \int l(x,\theta)p(\theta)d\theta + \int p(\theta)\log p(\theta)d\theta \\ &= \operatorname*{argmin}_p \int p(\theta)\log\frac{p(\theta)}{\exp(-l(x,\theta))}d\theta \\ &= \operatorname*{argmin}_p D\Big(p\|\exp(-l(x,\theta))\Big)\end{aligned}$$

Now from Proposition 1.5.1 it follows that this problem is minimised by

$$\hat{p} \propto \exp(-l(x,\theta))$$

This provides us with a non-degenerate distribution $\hat{p}$ that we are able to sample from. This idea can be taken one step further. Suppose we have a prior $\pi(\theta)$ over the parameters θ. Now instead of maximising the entropy of the parameters θ we could minimise the "distance" from $\hat{p}$ to the prior π. In order to minimise the distance we should minimise the Kullback-Leibler divergence. It is also interesting to note, that the entropy of θ is essentially the Kullback-Leibler divergence between its probability density function p and a "flat", constant prior, i.e. $H(\theta) = D_{KL}(p\|\text{ const})$.
In this case we would like to minimise the expected loss and the Kullback-Leibler divergence to the prior $\pi(\theta)$ together, our minimisation problem becomes

$$\begin{aligned}\hat{p}(\theta) &= \underset{p}{\operatorname{argmin}}\ \omega R(x,p) + D(p\|\pi) \\ &= \underset{p}{\operatorname{argmin}}\ \omega \int l(x,\theta)p(\theta)d\theta + \int p(\theta)\log\frac{p(\theta)}{\pi(\theta)}d\theta \\ &= \underset{p}{\operatorname{argmin}} \int p(\theta)\log\frac{p(\theta)}{\exp(-\omega l(x,\theta))\pi(\theta)}d\theta \\ &= \underset{p}{\operatorname{argmin}}\ D\Big(p\|\exp(-\omega l(x,\theta))\pi(\theta)\Big)\end{aligned}$$

And so just as above, it follows that it is minimised by

$$\hat{p} \propto \exp(-\omega l(x,\theta))\pi(\theta)$$

Note that we introduced a weight ω. This hyper-parameter allows us to choose how much weight we put on the loss function and how much on the prior.
This approach provides us with a distribution that we can sample from using MCMC Metropolis Hastings. We should note that the loss function is essentially taking the place of the log-likelihood in the traditional approach.

5.6 Algorithm

5.6.1 Introduction

In this subsection we introduce the algorithm for the network reconstruction, which allows non-linear relationships between the variables, non-Gaussian noise and cyclic structures. Our algorithm is an MCMC sampler, which samples from a loss function instead of the usual log-likelihood. The loss function (in this case it might be more convenient to view it as a cost function) is defined as the overall unexplained dependence left in the residuals. This raises some questions: how do we quantify the "dependence" in this context? and then how do we define "overall dependence"? To answer the first question is not too difficult: we may use an independence criterion (for example a distance covariance, Hilbert Schmidt independence criterion or signal to noise ratio criterion, for more detail see Section 3) to give us a numerical evaluation of how dependent a sample is. In our model we used the sum of all pairwise dependencies between the variables to measure the "overall dependency". A future extension could be to look at the mutual independence rather than the pairwise independence, though we currently do not have theoretical results to estimate the empirical mutual independence.

5.6.2 Loss Function Based on Independence Criterion

Lets assume we have observed data x which we assume was generated from a network on n nodes, i.e $x = (x_1, ..., x_n)$, where $x_i = (x_{i1}, ...x_{im})$, that is we have n variables and m observations. We furthermore assume that this data is being generated by the additive noise model defined in Section 5.2.2

$$x_i = \sum_{j \neq i} f(x_j, \beta_{i,j}) + \epsilon_i, \; \forall i = 1, ..., n$$

where ϵ_i are all independent and $\beta_{i,j} = \{\beta_{i,j,1}, ..., \beta_{i,j,k}\}$ are the k-dimensional vectors of parameters.
In the additive noise model we do not assume any particular distribution for the noise terms ϵ_i, therefore it is not possible to define a true Bayesian model and define a probability to observe data x given parameters β and equivalently we do not have a likelihood for our parameters β given the data x. What we can do, is define some meaningful loss function and sample the parameters using this loss function instead of the likelihood. We have introduced the general concept of the loss function in Section 1.4 and how to use it to sample with an MCMC in Section 5.5. Now it is time to discuss how it can be used in the context of the network inference using an Independence Criterion. Independence Criterion on its own provides us with a numerical value which can be interpreted as "how independent are the two samples". Given a graph on n nodes we would have n samples and so we would need to estimate $\binom{n}{2}$ pairwise independence criteria.
Let $r = (r_1, ..., r_n)$ be the residuals of this data after regressing it using the model with parameters $\theta = \{\theta_{i,j} : i, j = 1, ..., n\}$ and $\theta_{i,j} = \{\theta_{i,j,1}, ..., \theta_{i,j,k}\}$. We write the residuals as

$$r = g(x, \theta)$$

where $g = \{g_1, g_2, ..., g_n\}$ is an n-dimensional function with components g_i:

$$r_i = g_i(x, \theta) = x_i - \sum_{j \neq i} f(x_j, \theta_{i,j})$$

If we succeeded to find the correct parameters, that is $\theta_{i,j} = \beta_{i,j}$, for all $i, j = 1, ..., n$, the residuals r_i will be independent. We may use a loss function to estimate how independent the residuals are. One possible definition is:

$$\begin{aligned} l(x,\theta) &= \sum_{i=1}^{n}\sum_{j\neq i} \mathrm{IC}(r_i, r_j) \\ &= \sum_{i=1}^{n}\sum_{j\neq i} \mathrm{IC}(g_i(x,\theta), g_j(x,\theta)) \end{aligned}$$

Note that this can be interpreted as an absolute loss function. Here the "true" parameter value would be zero, i.e. true independence and $\mathrm{IC}(r_i, r_j)$ is the prediction using the decision θ. As all the independence criteria are difficult to interpret, it is easiest to justify an absolute loss function. On the other hand if we are more interested in penalizing strong dependences more than average dependence using L^p-norm for $p > 1$ might be better, i.e.

$$l(x,\theta,p) = \left(\sum_{i=1}^{n}\sum_{j\neq i} (\mathrm{IC}(r_i, r_j))^p\right)^{1/p}$$

5.6.3 Linear Model

Lets assume we have observations $x = (x_1, ..., x_n)$, where $x_i = (x_{i1}, ...x_{im})$, that is we have n variables and m observations. First we discuss the simplest possible model, i.e. a linear model. The model defined in Section 5.2.2 with a linear function f, is

$$f(x,\beta) = \beta x$$

The model is very similar to the Structural Equation Model (with the exception that we allow any noise ϵ distribution). We assume that the data is being generated by the additive noise model

$$x_i = \sum_{j\neq i} \beta_{i,j} x_j + \epsilon_i,\ \forall i = 1, ..., n$$

where ϵ_i are all independent of each other and of $\beta_{i,j}$ and $\beta_{i,j} \in \mathbb{R}$.
Using this model we can specify the loss function from the previous Section 5.6.2 as

$$\begin{aligned} l(x,\theta) &= \sum_{i=1}^{n}\sum_{j\neq i} \mathrm{IC}(r_i, r_j) \\ &= \sum_{i=1}^{n}\sum_{j\neq i} \mathrm{IC}\left(x_i - \sum_{l\neq i}\theta_{il}x_l, x_j - \sum_{k\neq j}\theta_{jk}x_k\right) \end{aligned}$$

5.6.4 Piecewise Linear Model

One possible extension of a linear model defined above is a piecewise linear model. This preserves the simplicity of linear functions but allows approximation of any continuous function. There are multiple parametrizations of a piecewise linear function, for example:

$$f_1(x;\alpha,k) = \sum_i \alpha_i(\min(x-k_i, k_{i+1}-k_i))_+$$
$$f_2(x;\beta,k) = \sum_i \beta_i(x-k_i)_+$$

here k represents the "knots", i.e. the points where the slope of function can change and α or β represents the slopes in-between two consecutive knots. Example of a piecewise linear function is provided in Figure 5.2, here the knots are $x = \{0, 0.2, 0.4, 0.6, 0.8, 1\}$ and the slopes are $\alpha = \{-1, -0.5, 0, 0.5, 1\}$ for the f_1 and $\beta = \{-1, 0.5, 0.5, 0.5, 0.5\}$ for f_2. Note that $\beta_i = \alpha_i - \alpha_{i-1}$, for $i = 2, 3, 4, 5$, this is no coincidence, as all piecewise functions can be easily re-parametrised for a different set up. We found the form of f_1 to be more convenient. The main reason is the interpretability of parameters α_i, it is precisely the slope in the segment i (while for example β_i is the difference of slopes in segment $i+1$ and i). For example if our MCMC provides us with the parameters $(-1, 0, 1)$ for an edge, we can interpret that as a quadratic relationship, while $(1, 0, 1)$ could be interpreted a cubic and $(0, 1, 0)$ as a sigmoid function.

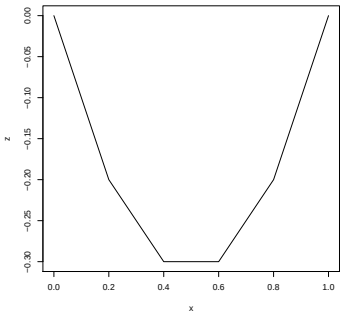


Fig. 5.2 Example of a piece wise linear function.

Using this model we can specify the loss function from Section 5.6.2 as

$$\begin{aligned} l(x,\theta) &= \sum_{i=1}^{n}\sum_{j\neq i} \mathrm{IC}(r_i, r_j) \\ &= \sum_{i=1}^{n}\sum_{j\neq i} \mathrm{IC}\left(x_i - f_1(x_j;\theta_{ij},k), x_j - f_1(x_i;\theta_{ji},k)\right) \end{aligned}$$

In this case we assume to use the same list of knots k for all variables $x_1, ..., x_n$. This is feasible if we normalize the data. If for some reason we would prefer to keep data on different scales, we may use a different set of knots k_i for each variable x_i. This may be even more appropriate if, for example, we assume only some of the relationships to be non-linear (we need a different number of knots depending on how non-linear the relationship is).

5.6.5 Prior

We intend to use a Bayesian approach and to have a full MCMC sampler. In the network inference case this requires a prior on both the models and the parameters. Lets suppose we have n observed variables and the space of possible models is all the possible networks on n nodes, then there are $2^{n(n-1)}$ possible models (we allow the edges in both directions). Let the model $G = (V_G, E_G)$ be a graph. Here are some possible priors on the models

1. $p_1(G) = 2^{-n(n-1)}$, this is a uniform prior on all the models
2. $p_2(G) = \binom{n(n-1)}{k} q^k (1-q)^{n(n-1)-k}$, this is a binomial prior, here k is the number of edges in the model M and $0 \leq q \leq 1$ is the a priori expected percentage of edges

Most of the time we do not have any prior knowledge (for example from human experts) about the possible structure of the network. In this case it is natural to assume a uniform prior on all possible models. On the other hand we might expect a certain density of the network (for example from other networks in the same field), then it might be a better idea to use a prior distribution with a mode at a network with $\approx qn(n-1)$ number of edges. Such a prior can also serve as a tool to force a sparse solution.
After choosing a prior for the models, we still need to define a prior for the parameters of the model. Here are some possibilities:

$$
\begin{aligned}
q_1(\theta; G, p) &\propto \prod_{i,j:\{i,j\}\in E_G} \exp\left(-\|\theta_{ij}\|_p\right) \\
q_2(\theta; G, \sigma) &\propto \prod_{i,j:\{i,j\}\in E_G} \exp\left(-\frac{\|\theta_{ij}\|_2^2}{2\sigma^2}\right) \\
q_3(\theta; G, f) &\propto \prod_{i,j:\{i,j\}\in E_G} \exp\left(-f\left(\theta_{ij}\right)\right)
\end{aligned}
$$

Recall that θ_{ij} is a k-dimensional vector, where k is the number of parameters required to parametrize an edge. For the first prior q_1 we would need to specify the norm parameter $p \geq 1$, this parameter can be used to have some control on the parameters θ_{ij}. Lets recall that

$$\|\theta\|_p = \left(\sum_{l=1}^{k} \theta_l^p\right)^{1/p}$$

Now lets consider a constrained minimisation problem:

$$\min_{\theta} \|\theta\|_p$$
$$\|\theta\|_1 = C$$

for some constant C. This is minimised by $\theta_l = C/k$ for all l. So choosing p larger than 1 forces the parameters θ_{ijl} for the edge $i \to j$ to be closer to each other. If for example we choose the piece wise linear parametrisation f_1 defined in Section 5.6.4, this will prioritize a linear relationship (that is all the slopes in the piecewise linear parametrisation being the same). The larger the parameter p the greater will be the cost of choosing different slopes.
The second prior q_2 is just a joint Gaussian centred at zero. In this case we can choose the diffusion parameter σ. The smaller the σ parameter, more concentrated around zero is the prior probability and larger the penalty for choosing non-zero parameters.
The third prior q_3 is just a general case covering both q_1 and q_2, we can choose any function f. One possible option could be

$$f(\theta) = \omega_1 \left(\|\theta\|_1\right)^{p_1} + \omega_2 \left(\mathrm{Var}\left(\theta\right)\right)^{p_2} \tag{5.6.1}$$

We have 4 parameters we can adjust in order to enforce the desired properties of relationships between variables in the network. For example increasing ω_2 will force MCMC towards linear relationships, as if a piecewise linear function with parameter θ represents different slopes then the variance of parameter θ will be non-zero. Parameter ω_1 will determine the overall size of parameters (in the sampler ω represents the weight we put on the loss function with respect to the prior), while the power parameters p_1 and p_2 determine how strongly we punish the small or large parameters.

5.6.6 Sampling Distribution

We combine the above ideas to define a full distribution to sample from. We chose to use the prior p_2 for the models as it gives more flexibility with respect to the sparsity

of the network and the prior q_3 for the parameters as it allows us to flexibly penalize non-linear relationships. Combining it all we get

$$\begin{aligned} p_{G,\theta|x} &\propto \exp\left(-\omega l\left(X,\theta\right)\right) p\left(G\right) p\left(\theta;G\right) \\ &\propto \exp\left(-\omega \sum_{i=1}^{n}\sum_{j\neq i} \mathrm{IC}(r_i,r_j)\right) p\left(G\right) \frac{1}{C(|E_G|)} \prod_{i,j:\{i,j\}\in E_G} \exp\left(-f\left(\theta_{ij}\right)\right) \\ &\propto \exp\left(-\sum_{i=1}^{n}\sum_{j\neq i} \left(\omega \mathrm{IC}(r_i,r_j) - \mathbf{1}(\{i,j\}\in E_G) + \log(C(|E_G|))f\left(\theta_{ij}\right)\right)\right) p(G) \end{aligned} \tag{5.6.2}$$

If we use f as defined in Equation 5.6.1, we have 5 parameters for control: ω, ω_1, ω_2 and p_1, p_2. Parameter ω has essentially the same purpose as the parameters ω_1 and ω_2 therefore we are keeping it fixed and changing only the ratios $\frac{\omega_i}{\omega}$, $i = 1, 2$. Note that in the parallel tempering approach if we keep the ratios $\frac{\omega_i}{\omega}$ fixed and change ω this is equivalent to adjusting the temperature.

5.6.7 Hyperparameters

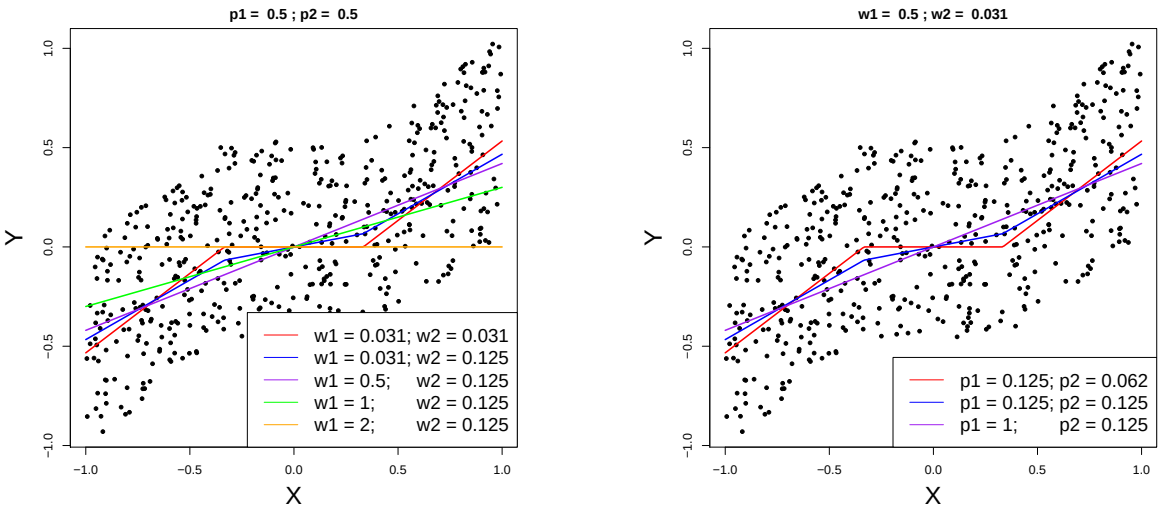
As we have discussed above our sampling distribution has 4 hyperparameters that have to be set, namely:

- ω_1 determines the overall importance of the loss function (or overall penalty for a relationship)
- ω_2 determines how strongly a non-linear relationship is penalized
- p_1 determines how strongly are penalized the small and large parameters for the linear term
- p_2 determines how strongly are penalized the small and large parameters for the non-linear term

Lets consider an example of a simple model $X \to Y$, where data is sampled from the model

$$\begin{aligned} X &\sim U[-1,1] \\ Y &\sim f_1(X;\alpha=\{0.8,0,0.8\},k=\{-1,-1/3,1/3,1\}) + U[-0.5,0.5] \end{aligned}$$

Here f_1 is a piece-wise linear function as defined in 5.6.4. We consider parameters in the following ranges: $\omega_1, \omega_2 \in \{2^{-6}, 2^{-5}, ..., 2^4\}$ and $p_1, p_2 \in \{2^{-5}, 2^{-4}, ..., 2^0\}$. Depending on the hyper parameters, 5 different types of outcomes will be most likely as demonstrated in Figure 5.3a: true non-linear relationship (demonstrated by the red line), penalized non-linear relationship (all piece-wise linear gradients are closer to each other as to reduce the variance among them; demonstrated by the blue line), best linear approximation (all gradients for piece-wise linear elements are equal, i.e. zero variance; demonstrated by the purple line), linear approximation with gradient being reduced (demonstrated by the green line) and no relationship at all (it is more expensive to have any non-zero parameter than the full value of dependence criterion; demonstrated by the orange line). Here we demonstrate only some examples of the best solutions. As we continuously vary the hyper-parameters the transition between them is continuous as well. We chose to fix the power parameters p_1 and p_2 and show different solutions for varying ω_1 and ω_2, but same result can be achieved by fixing ω's and varying the power parameters as seen in Figure 5.3b. It is important to note that by just varying the power parameter we cannot enforce the zero solution (we want to keep the power parameters smaller than one because it allows to penalize small values more).



(a) Varying parameters ω_1 and ω_2.

(b) Varying parameters p_1 and p_2.

Fig. 5.3 The best solutions for different parameters.

5.6.8 Sampler

For our sampler we are using the loss function defined in Equation 5.6.2. Based on the discussion in Section 5.6.7 we set the values of the hyperparameters to be: $p_1 = p_2 = 0.5$, $\omega_1 = \omega_2 = 2^{-4}$ and $\omega = 200$. ω was chosen to provide a good acceptance rate for the jumps inside the highest temperature chain (around 0.2).
The loss function involves calculating $\text{IC}(r_i, r_j)$ for all $i, j = 1, ..., n$. If we restrict our algorithm to only making moves "add", "remove", "revert" or "update" an edge, then at most two residuals will change (two in the case of changing the directionality of the edge, otherwise only one). This implies that we actually do not have to recalculate all the pairwise independence criteria. For example if we updated (changed the parameters for) edge $X_i \to X_j$, then only the residuals r_j has changed, so we need to calculate only $\text{IC}(r_i, r_j)$ for all $i \neq j$. This is a significant simplification for large n as now we only need to calculate order of n independence criteria rather than order of n^2.

5.7 Results

5.7.1 Simulating Data with Cycles

In order to be able to generate data from the networks with cycles we first need to discuss what does a cycle in a network represent? Suppose we have n-dimensional variable $X = (X_1, ..., X_m)$ which is generated by a dynamical system. Let t_k for $k = 1, 2, ...$ be arbitrary spaced time points, then the state of the system at time t_k (denoted as $X[t_k]$) is the function of the state of the system at time t_{k-1}, i.e.

$$X[t_k] = f(X[t_{k-1}], \epsilon), \text{ for all } k = 1, 2, ...$$

here f is n-dimensional function, i.e. $f = (f_1, ..., f_n)$ and $X_i[t_k] = f_i(X_1[t_{k-1}], ..., X_n[t_{k-1}], \epsilon)$, for all $i = 1, ..., n$ and $k = 1, 2, ...$ We think of ϵ as some initial conditions that determines the evolution of the dynamical system.
Example An example of such a dynamical system is a structural equation model

$$X_k = WX_{k-1} + \epsilon$$

We provide two interpretations of a cyclic relationship in a network. First interpretation is given by Fisher (1970). He suggested that there is a long observation period (t_{k-m}, t_k) and much shorter reaction period (t_{k-i-1}, t_{k-i}). The observed value $x[t_k]$ at time t_k is a mean over the observation period, i.e.

$$x[t_k] = \frac{1}{m}\sum_{i=0}^{m} f(X[t_{k-i}, \epsilon]), \text{ for all } k = 1, 2, ...$$

Using the SEM example we can rewrite this as

$$x = \frac{1}{m}\sum_{i=0}^{m} W^i \epsilon$$

Another interpretation is that the observed value is an observation from the converged dynamical system, i.e. $x = x[t_k]$ for t_k such that $x[t_k] = x[t_{k-1}]$. Using the SEM example (given that $I - W$ is invertible)

$$x = (I - W)^{-1}\epsilon$$

Note that if $I - W$ is invertible, then $\lim_{m\to\infty} \sum_{i=0}^{m} W^i = (I - W)^{-1}$, so both interpretation yield very similar result and therefore very similar approaches to generating data from a dynamical system with a cycle, but we follow the second one. First we generate the noise term for each observation (which represents a different patient or a single cell) i as $\epsilon_i \sim p$ from some noise distribution p. Initialize the dynamical system at $X_0 = \epsilon$. Then we run the dynamical system till it converges (usually 100 iterations is enough): $X_i = f(X_{i-1}) + \epsilon$. Note that function f has to be chosen carefully for the dynamical system to converge, in the SEM example we must choose the weight matrix W so that $I - W$ would be invertible. The pseudo code for the algorithm is provided in Algorithm 5.

5.7.2 Simulated Examples

In this section we discuss the results of our algorithm on small networks. We used a very small and slightly bigger examples with a 2-cycle, an example with 4-cycle and finally an example with 2 2-cycles. First we consider only the linear relationships. Finally we consider some examples with non-linear relationships.
For all these examples we ran 6 chains at temperatures varying from 1 to 1/4. The ratio between temperatures was chosen to provide a good acceptance rate for swaps between two consecutive chains. We ran each chain for 10^6 iterations and we allowed the chains to swap every 10^3 iterations, i.e. 10^3 times in total (for a detailed explanation of tempered MCMC, see Section 2.5). It is always difficult to determine whether the MCMC has converged, but as in this case we know the correct networks and their parameters we conclude that this is sufficient, we provide an example MCMC run for each example and it is clear that the chains has converged.

Data: n the number of observations
m the number of variables
$p = (p_1, ..., p_m)$ the noise term distribution
$f = (f_1, ..., f_m)$ the functional relationships
Result: Data sample
for $i = 1 : n$ **do**
  for $j = 1 : m$ **do**
    $\epsilon_{ij} \sim p_j$ generate the noise term for each variable
    $X_{ij} = \epsilon_{ij}$ initiate the starting point for the dynamical system
  end
  for $k = 1 : 100$ **do**
    for $j = 1 : m$ **do**
      $X_{ij} = f_j(X_{i1}, ..., X_{im}) + \epsilon_{ij}$ run the dynamical system till it converges
    end
  end
end
return $X = X_{ij}$ for $i = 1, 2, ..., n$ and $j = 1, ..., m$.

Algorithm 5: Algorithm to simulate data from a network with a cycle.

Simple network with a cycle

The first example (see Figure 5.4) is a network on 4 nodes with 4 edges, all relationships are linear, noise is uniform. It has one cycle with 2 nodes. Figure 5.5a shows the MCMC chain at the lowest temperature (we assume that this is the chain that is sampling from the "true" distribution), the red solid line shows the true parameter value and the red dashed line show the mean of the MCMC sample. We see that the chain converged very well and is nicely sampling the parameters in the neighbourhood of the true parameters. Figure 5.5b shows the MCMC chain at the highest temperature, we see that it is exploring the full region of possible parameters and therefore provides a good mixing. Though we should point out that because it is moving around so much it does not provide good estimates of the parameters. Finally in Figure 5.6 we present the MCMC chain using 3 knots. As all the relationships are linear we expect the piece-wise linear function to have tangents of the piece-wise linear segments very close to each other. We observe that the chain has converged well and recovered the linear relationships between variables, only the edge $X_1 \to X_3$ shows a little bit of non-linearity.

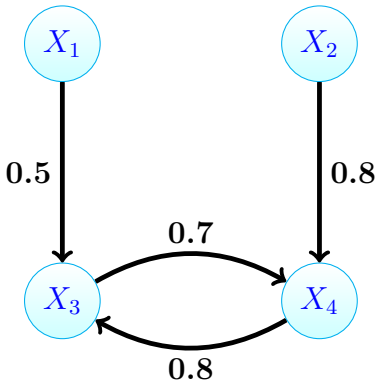
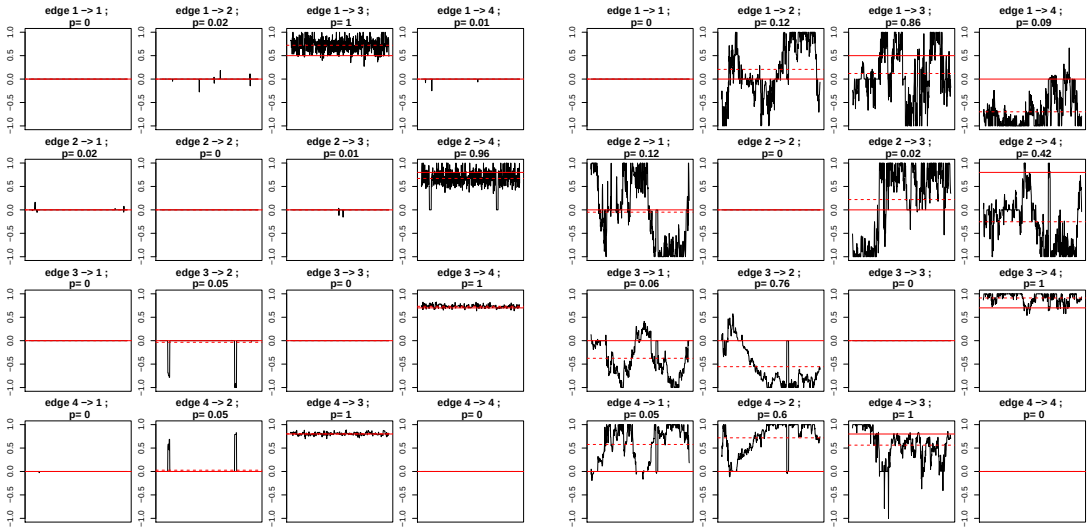


Fig. 5.4 Small example with a cycle.



(a) MCMC simulation for Figure 5.4 using 2 knots, chain at the temperature $t = 1$.

(b) MCMC simulation for Figure 5.4 using 2 knots, chain at the temperature $t = 0.25$

Fig. 5.5 Small example with 4 nodes and 4 edges, containing a cycle. Red solid line represents the true parameter value, red dotted line the mean MCMC sample value.

Bigger network with a cycle

The second example (see Figure 5.7) is a slightly bigger (but still small) graph on 6 nodes with 8 edges. We are still using the linear relationships and non-Gaussian noise. In Figures 5.8a and 5.8b we provide two sample MCMC runs. The first run allows only 2 knots per edge, that is only linear relationships are allowed, while the second run allows 3 knots per edge. As before we observe that MCMC chains converged well to the true parameter values. The MCMC run in Figure 5.8b has both the parameter values for each edge close to each other, so we may conclude that the penalty for non-linearity was chosen well.

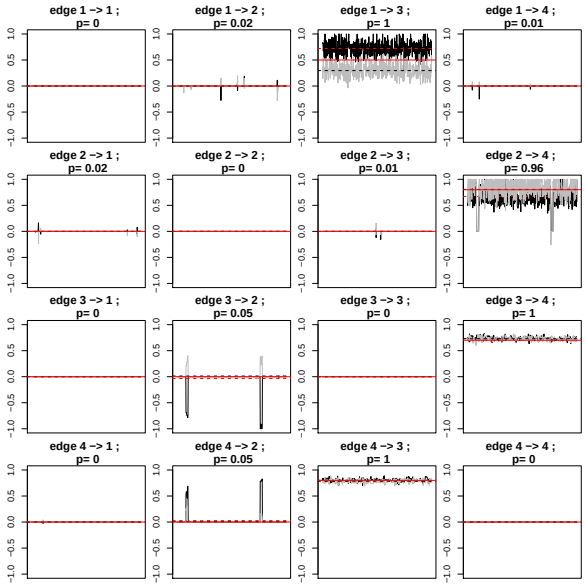


Fig. 5.6 MCMC simulation for Figure 5.4 using 3 knots.

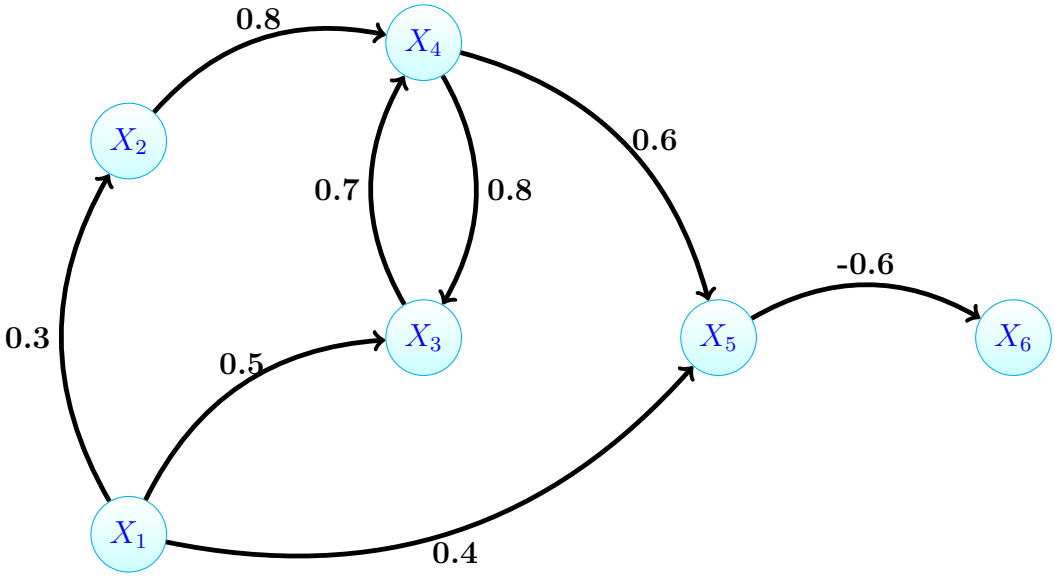
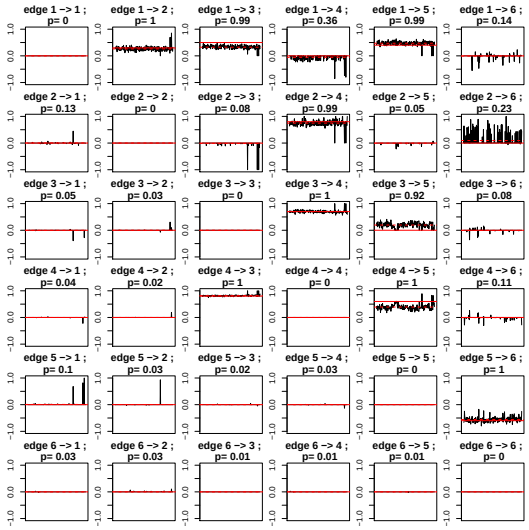


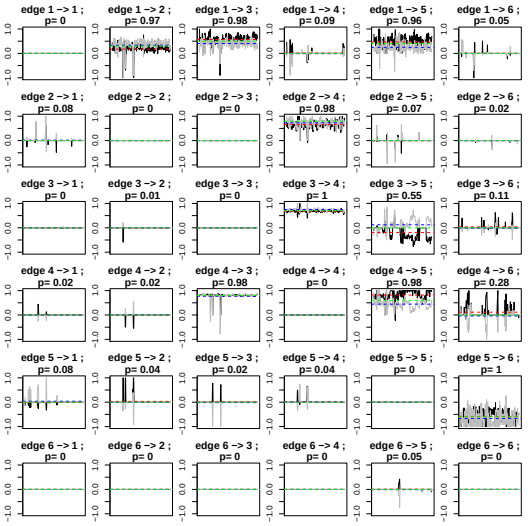
Fig. 5.7 Second small example with a cycle.

Network with a bigger cycle

The third example (see Figure 5.7) is a network on 6 nodes and 7 edges, containing a "big" cycle of 4 nodes. In Figure 5.10 we provide a sample MCMC run. We note that not only did it correctly identified the cycle, but also the directionality of the cycle. All the correct edges has the posterior probabilities above 0.98 and edges that are not in the network has probabilities of at most 0.08.



(a) MCMC simulation for Figure 5.7 using 2 knots.



(b) MCMC simulation for Figure 5.7 using 3 knots.

Fig. 5.8 Small example with 6 nodes and 8 edges, containing a cycle. Red solid line represents the true parameter value, red dotted line the mean MCMC sample value.

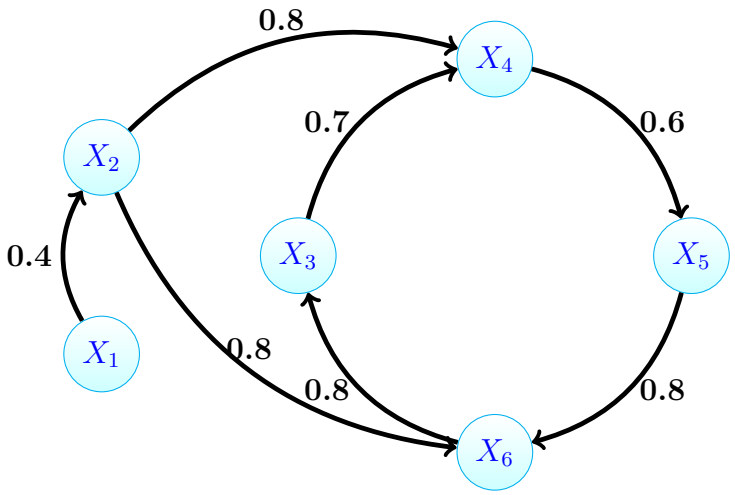


Fig. 5.9 Example with 6 nodes and 7 edges, containing a "big" cycle of 4 nodes: $X_3 \to X_4 \to X_5 \to X_6 \to X_3$.

Network with two cycle

The fourth example (see Figure 5.11) is a network on 7 nodes with 11 edges, containing two cycles. In Figure 5.12 we provide a sample MCMC run. All the correct edges has the posterior probabilities of at least 0.9, but this time some of the edges that are not in the network has quite high posterior probabilities, e.g. $p(X_5 \to X_1) = 0.57$, $p(X_1 \to X_6) = 0.26$, $p(X_7 \to X_1) = 0.26$. Though we should note that even though

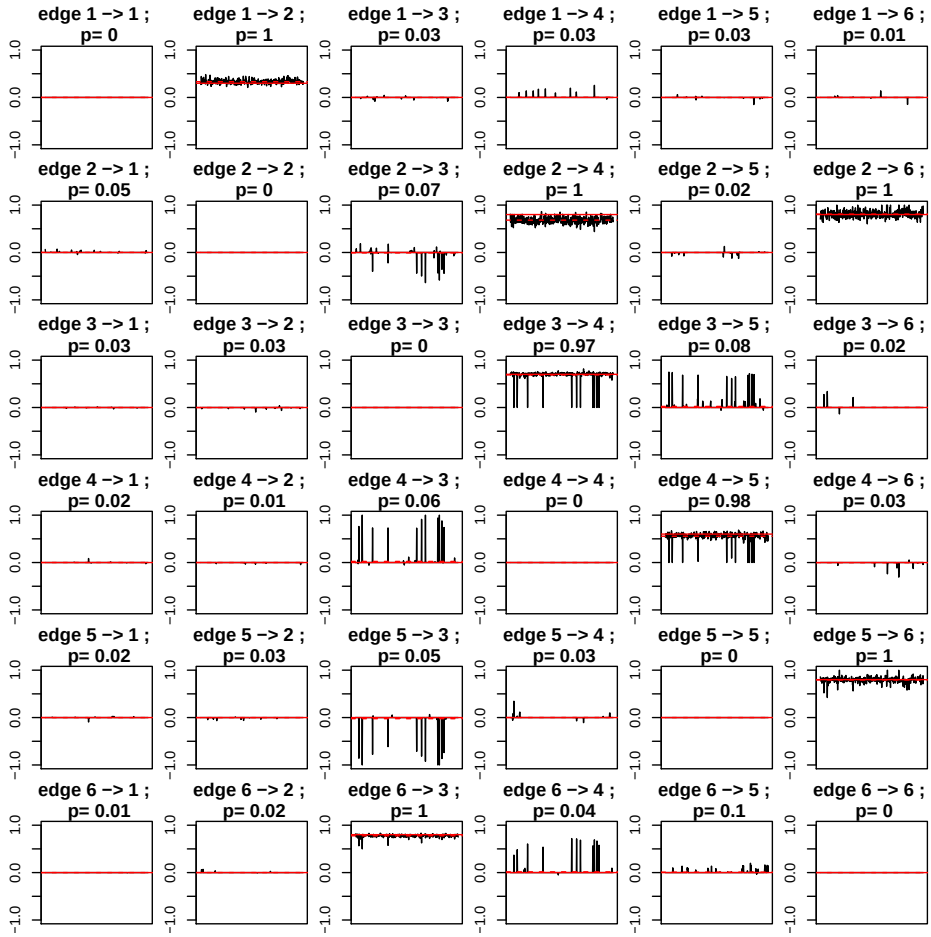


Fig. 5.10 MCMC simulation for Figure 5.9. Red line represents the true parameter value, red dotted lines the mean MCMC sample value.

these edges appear in the model quite often, their mean parameter values are very close to zero. This suggests that when dealing with real data, where we do not know the true underlying network, we should consider not only the posterior probabilities of the edges but the magnitude of their parameters as well.

Example with non-linear relationship

The last example (see Figure 5.13) is a network on 4 nodes with 3 edges containing a cycles and non-linear relationship $X_1 \to X_4$, i.e. $X_4 \sim f_1(X_1; \alpha = (-0.8, 0, 0.8)) + \epsilon$. In Figure 5.14 we provide a sample MCMC run. We observe that both linear relationships were identified as such, i.e. all 3 parameters are close to each other and the true parameter value. The non-linear relationship was also identified correctly with just a slightly minimized parameter values.

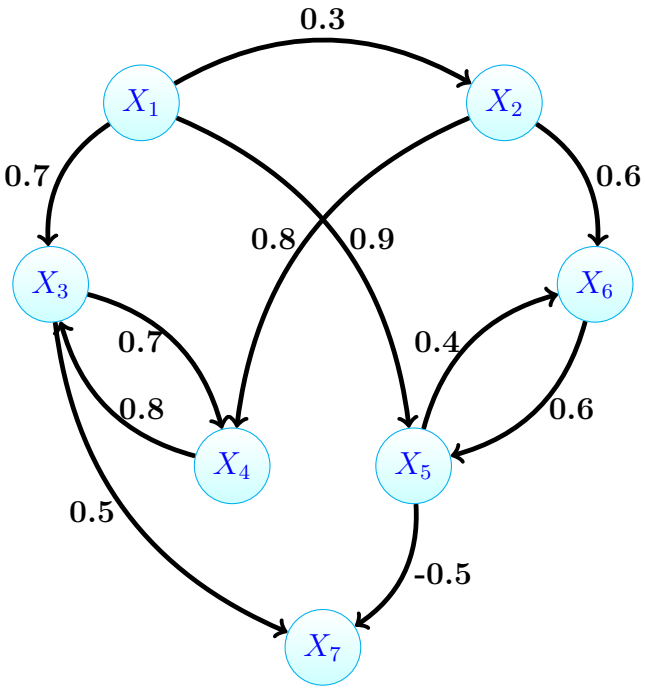


Fig. 5.11 Example with 7 nodes and 11 edges, containing two cycles: $X_3 \leftrightarrow X_4$ and $X_5 \leftrightarrow X_6$.

Discussion

In this subsection we considered 5 small simulated examples, each exhibiting a trait of interest: a cycle, big cycle, two cycles or non-linear relationships. When the true relationship between the variables is linear and the piece-wise linear function is used, algorithm correctly identifies relationship to be linear and uses the tangents for all the piece-wise linear elements very close to each other. When the relationship is truly non-linear the algorithm is capable of finding correct parameters for the non-linear function (granted that we were using the correct model, i.e. the piece-wise linear relationships). We also observed that as the networks are getting larger and more complicated we may have to rely not only on the posterior probabilities of the edges, but on the parameter estimates as well. Edge being in the model with relatively high probability but having a parameter very close to zero is a good indication that the edge actually should not be in the final model. This raises a question that maybe a more sophisticated penalty terms for edges with low parameters should be used.

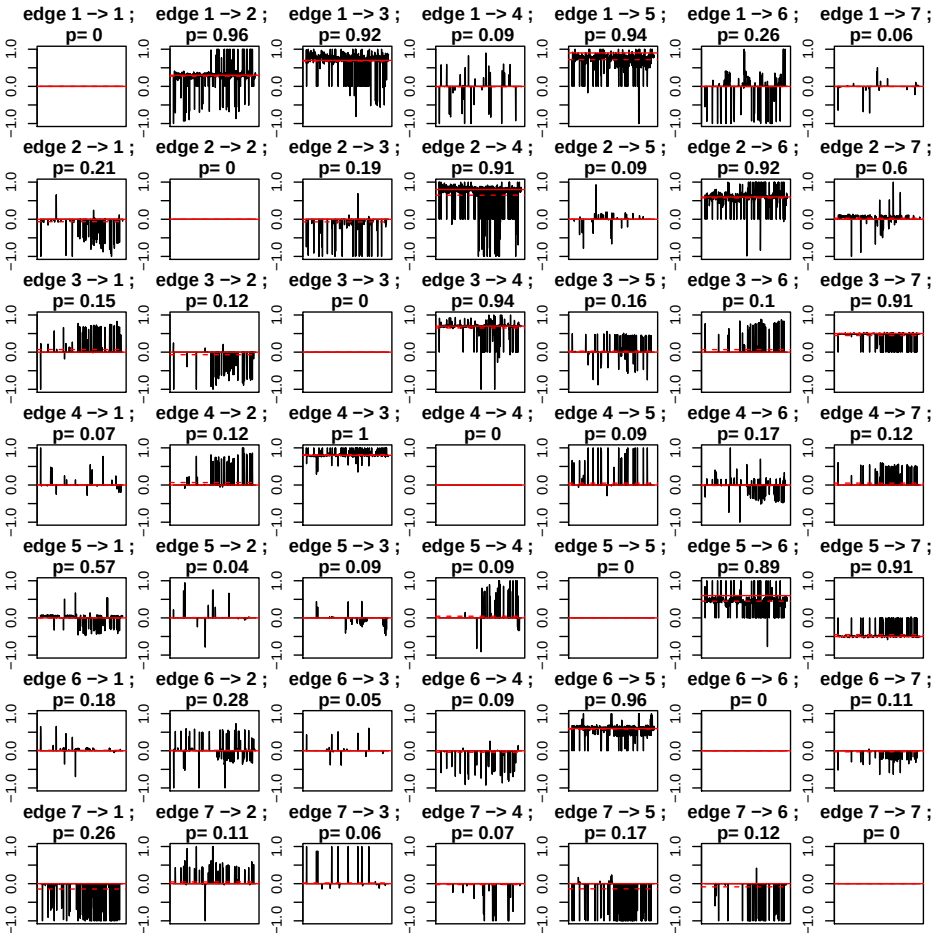


Fig. 5.12 MCMC simulation for Figure 5.11. Red line represents the true parameter value, red dotted lines the mean MCMC sample value.

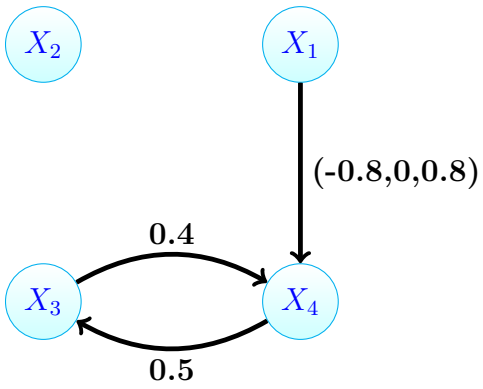


Fig. 5.13 Second small example with a cycle and non-linear relationship.

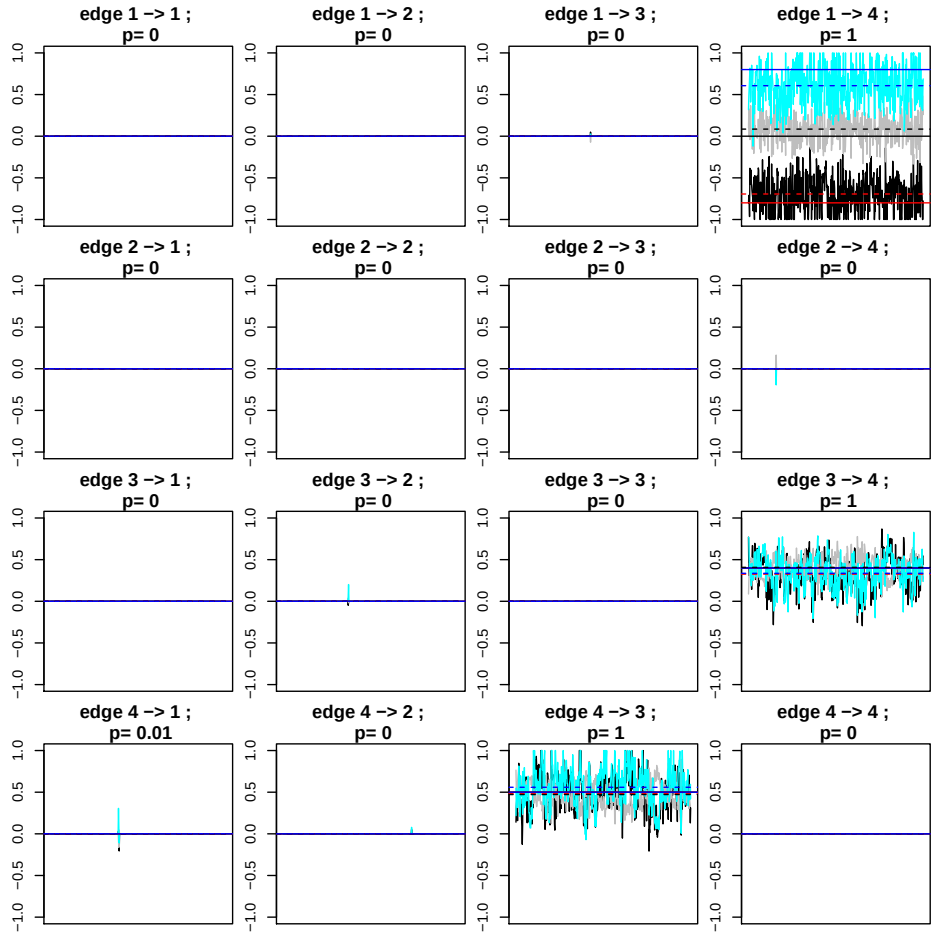


Fig. 5.14 MCMC simulation for Figure 5.11. Solid lines represents the true parameter values, dotted lines the mean MCMC sample values.

5.7.3 Single-Cell Data

In this section we present the results of the MCMC sampler using an independence criterion based on a loss function on the single-cell data from (Sachs et al., 2005). As in the previous Chapters 2 and 4 we first used the 100 simulated datasets simulated from dataset 8. We ran the MCMC for 10^6 iterations using 6 chains at temperatures ranging from 1 to 1/4. We used 2 knots, i.e. linear relationships. The current approach enables us to find directions of edges as well as cycles therefore we are presenting two different ROC curves: one for an undirected graph (skeleton) and one for directed graph. ROC curves are shown in Figure 5.15. The area under the ROC curve for the skeleton is $0.831/pm0.04$, this is better result than all the kPC variants which gave areas under the ROC curve from 0.8 to 0.81, though these results are within one standard deviation. A better metric is the number of times MCMC with loss function

outperformed the kPC variants: 71 for dPC, 77 for kPC-Resid, 72 for kPC-Clust, 70 for SNR-PC and 95 for original PC. The area under the ROC curve for the directed graph is 0.777 ± 0.04, as expected it is slightly lower than the undirected version. We provide the consensus network for the single-cell data in Figure 5.16. We note that the algorithm found all the relationships with non-negligible signal to noise ratio as well as correctly identified the cycle $PIP2 \leftrightarrow PIP3$. The other cycle that we would expect to find is $PIP2 \leftrightarrow PKC$ but this is expected as the signal between these variables is very small, all 4 independence tests (dCov or HSIC criterion using permutation or gamma approximation tests) give quite large p-values; they vary from 0.1 to 0.15. The only two wrong edges, namely $P38 \rightarrow PKC$ and $AKT \rightarrow PKA$ are correct edges but with a reversed directionality. This suggests that the relationships $P38 - PKC$ and $AKT - PKA$ might not be well approximated by a linear function. Finally we provide the results of running the MCMC with loss function on the single-cell dataset 8 in Figure 5.17. Based on the observations from simulated datasets we used non-linear relationships with 4 knots. Results are even better than for the simulated dataset but it is well within one standard deviation from the mean. Recall from previous chapters that the simulated data was always easier to deal with and was giving larger area under the curve than the experimental one. We conclude that using non-linear relationships may help significantly when inferring network from the experimental data.

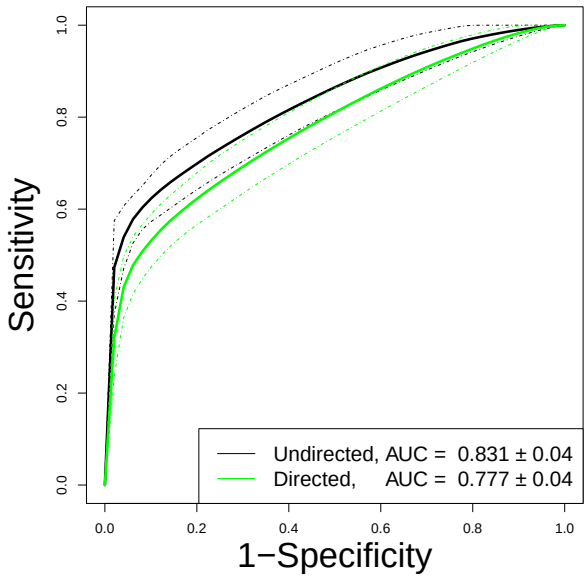


Fig. 5.15 ROC curves for the datasets simulated from the single-cell dataset 8. MCMC with loss function using linear relationships.

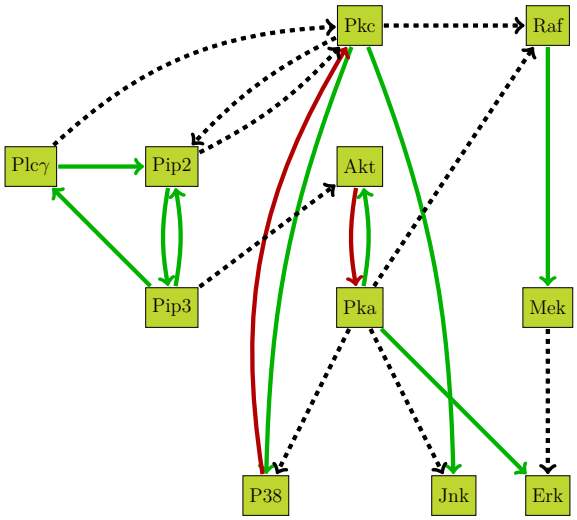


Fig. 5.16 Network for the datasets simulated from the single-cell dataset 8. Green are the correctly identified edges, red are wrongly identified edges and dashed black are the edges that algorithm did not find.

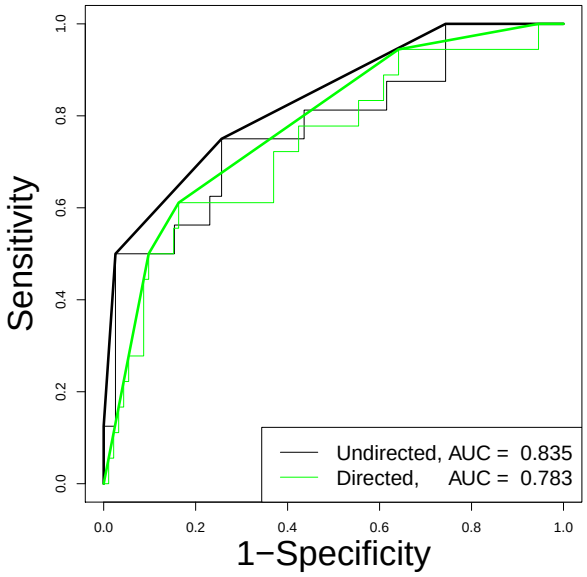


Fig. 5.17 ROC curves for the single-cell dataset 8. MCMC with loss function using non-linear relationships.

5.7.4 Schistosomiasis Dataset

Finally we present the results on the Schistosomiasis dataset. As we have discussed in previous chapters we do not know the true underlying network for this dataset so evaluating results is difficult. We start with linear relationships. We present the output of 4 sample MCMC runs in Figure 5.19. All four MCMC runs produced

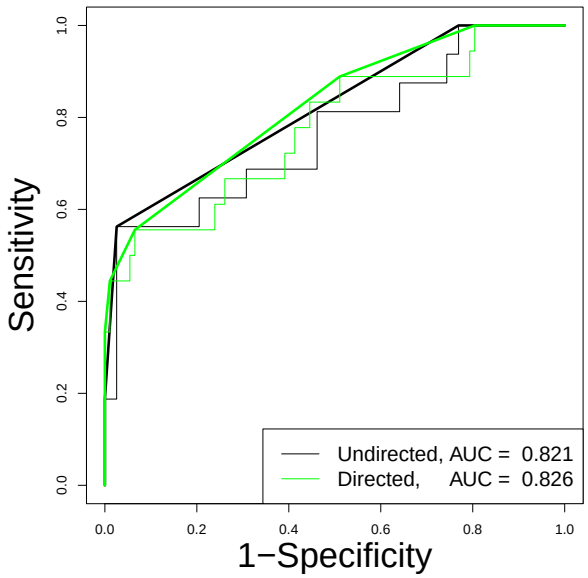


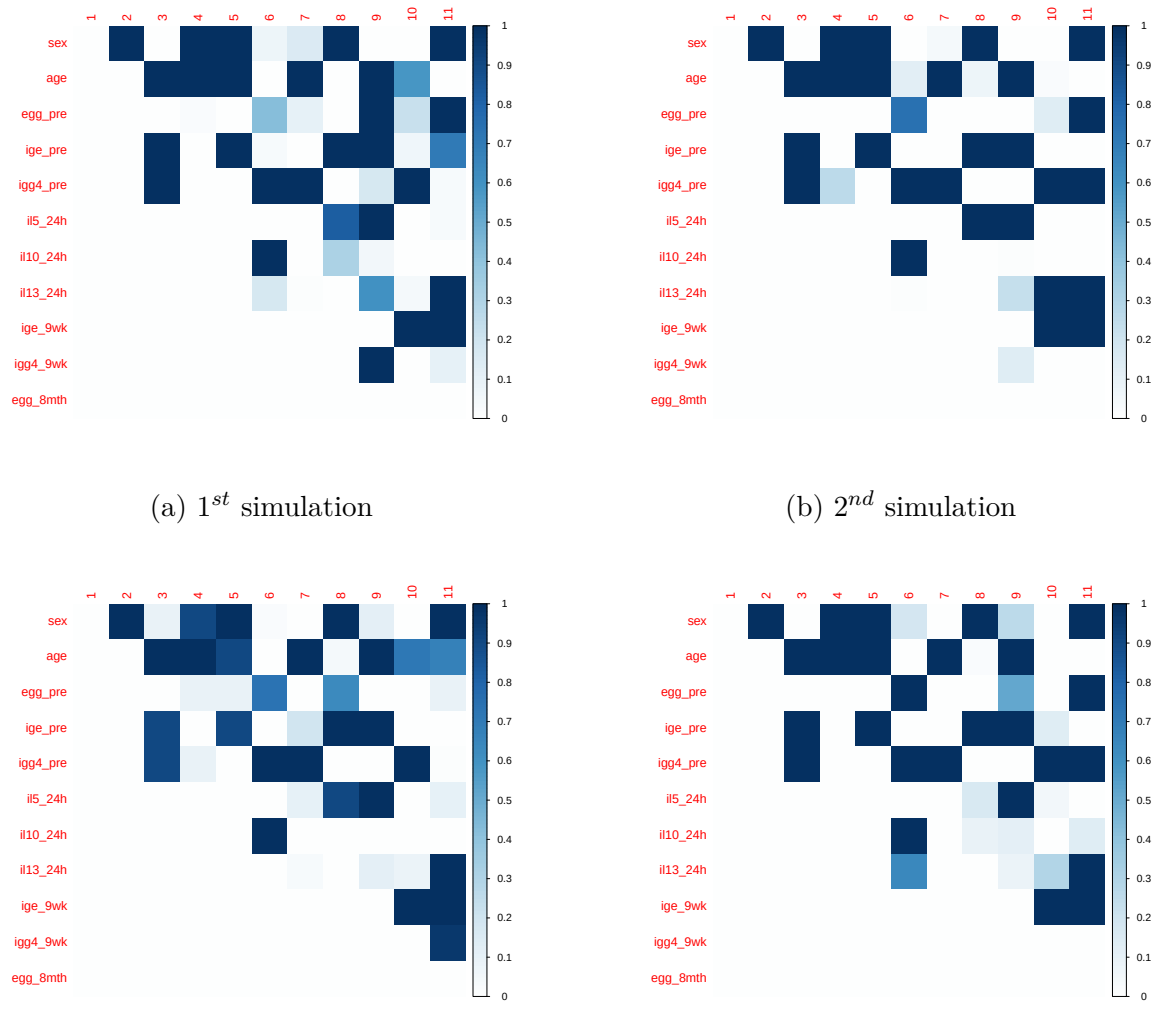
Fig. 5.18 ROC curves for one dataset simulated from the single-cell dataset 8. MCMC with loss function using non-linear relationships.

slightly different posterior distributions for edges being in the network. But as we have learned from small simulated examples, when network gets relatively big looking at the probabilities of edges being in the network might not be sufficient. Therefore we present the actual mean parameter values for each of the edge in Figure 5.20. The mean parameter values for all four chains are essentially identical. We conclude that the MCMC chains converged successfully.

5.8 Discussion

The aim of this chapter was to explore the possibility of combining the probabilistic independence criteria with the MCMC sampling techniques for network inference. To achieve this we used ideas taken from Bissiri et al. (2016), this allowed us to come up with an MCMC sampler that samples from a distribution defined by a general loss function rather than a likelihood. Choosing the loss function to be the sum of pairwise independence of the residuals allowed us to naturally incorporate the probabilistic independence criterion.

Our algorithm dealt with small examples containing cycles and non-linear relationships very well. It was both able to perfectly reconstruct the skeleton of the network and find the directionality of edges and cycles. As expected the convergence gets worse as networks get larger, we may end up with edges that have strictly positive posterior



(a) 1^{st} simulation

(b) 2^{nd} simulation

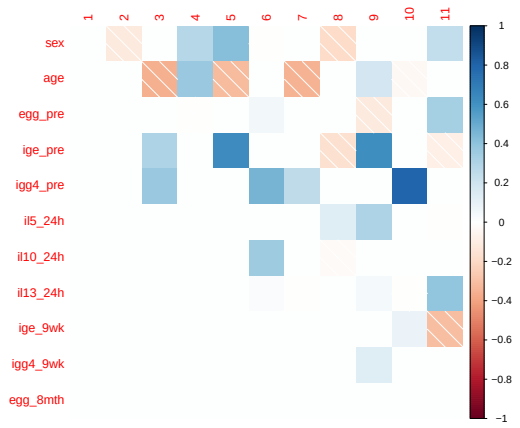
(c) 3^{rd} 25 simulation

(d) 4^{th} simulation

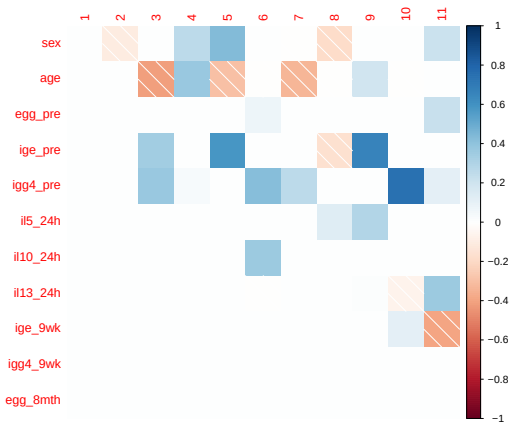
Fig. 5.19 The Schistosomiasis network adjacency matrix found by our algorithm. Matrix M element m_{ij} represents the probability that the i^{th} node is a parent of the j^{th} node.

probability but negligible parameters. This suggests that if we want to use only the posterior probabilities of the edges and to not rely on their parameters we would have to introduce new penalty terms for "empty" edges, i.e. edges with parameters very close to zero.

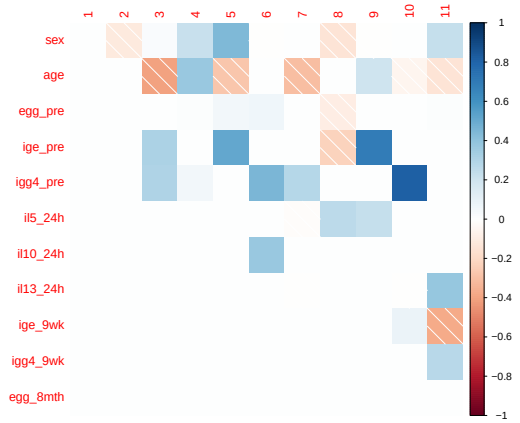
Next we explored the performance of our algorithm on experimental single-cell datasets. We first attempted to use only the linear relationships as it significantly reduces the number of parameters that have to be estimated. The results were slightly better than the ones produced by the greedy search type kPC algorithm from Section 4 or the



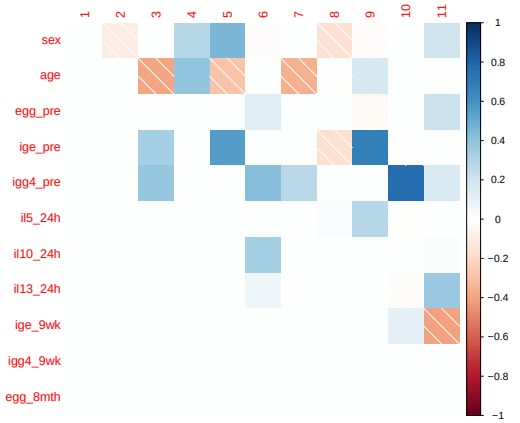
(a) 1^{st} simulation



(b) 2^{nd} simulation



(c) 3^{rd} simulation



(d) 4^{th} simulation

Fig. 5.20 The Schistosomiasis network adjacency matrix found by our algorithm. Matrix M element m_{ij} represents the size of the effect of the i^{th} node on the j^{th} node.

MCMC sampler using discretized data from Section 2. We observed that all the false negatives (edges that are supposed to be in the network but were not found by the algorithm) were the edges with very low signal to noise ratio, while most of the false positive (edges found by the algorithm that are not truly there) were the correct edges that were inverted. This suggested that linear approximation might not be sufficient. We continued by using the non-linear relationships, we used 4 knots which is sufficient to approximate the most common relationships, i.e. linear, quadratic, cubic, sigmoid,

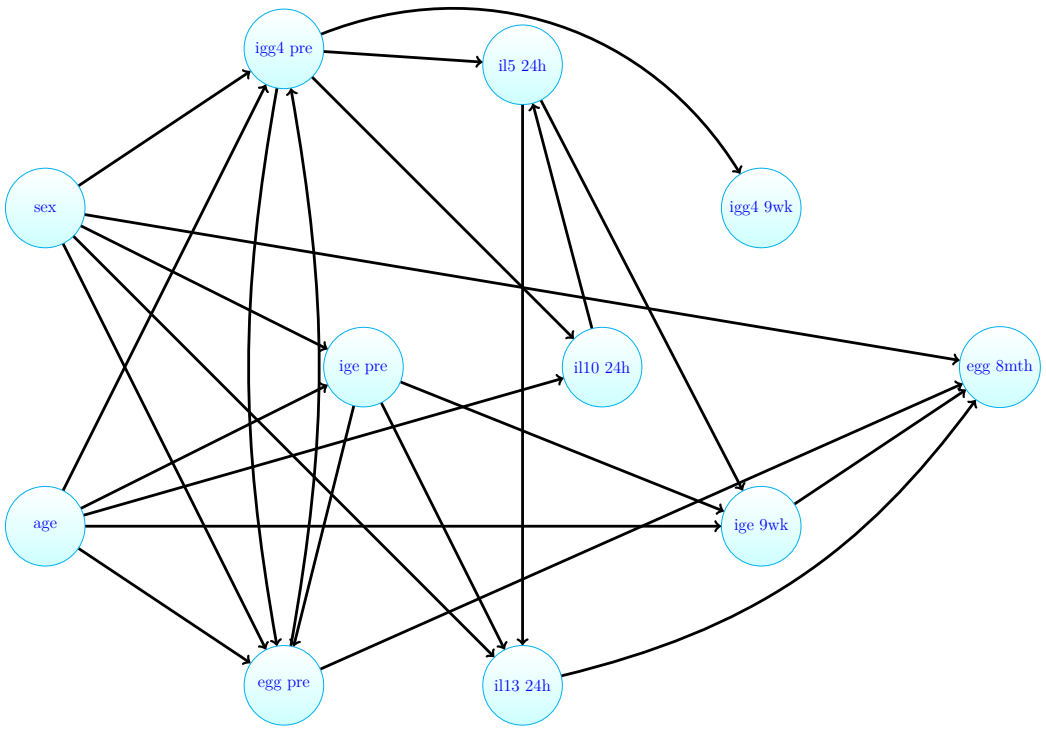


Fig. 5.21 Network for the Schistosomiasis dataset; only the edges with probability above 0.5 and parameters above 0.1 in absolute value are present.

etc. This produced similar results in terms of the skeleton, but significantly better results in terms of the directed networks.

Finally we explored the Schistosomiasis dataset. We do not have a consensus network in this case therefore it is difficult to evaluate results. Using linear relationships gave a consistent network which is a good estimate that the MCMC has converged. Allowing the non-linear relationships created more convergence issues. It suggests that for a dense network like Schistosomiasis with non-linear relationships the probability surface is significantly more complicated and some different approach than the parallel tempering might be required.

Our work provides empirical evidence that it is possible to combine a probabilistic independence based loss function with an MCMC sampler for network inference. This approach finds network well but is computationally slow.

5.8.1 Model Limitations

In this chapter we discussed the algorithm of inferring network by MCMC sampling from a loss function based on an independence criterion. We have shown this to be a

very flexible approach that can deal with non-linear relationships between variables, non-Gaussian noise and loops. Despite the flexibility and the good performance, the main limitation of this algorithm is its computational complexity and therefore its speed.

We have discussed the problems that arise in network inference when using MCMC based structure search in Section 2.7.1. Introducing a semi-parametric model such as MCMC sampling from a loss function based on an independence criterion adds an additional layer of complexity. Now we need to sample parameters as well as structure. Empirically we determined that approximately 20 parameter update proposals to every structure update proposal works quite well. This estimate is for linear and two piece piece-wise linear relationships between variables and for networks under 12 nodes. Using a more complex relationships (e.g. with more than two piece-wise linear components) might require more parameter update proposals per one structure update proposal to give an efficient sampling. Sampling parameters as well as structures results in having to run MCMC chain at least 20 times longer to achieve same result as in the closed form Discrete Bayesian Network case of Chapter 2 (there we were sampling only structures).

Second issue is the lack of closed form for the likelihood. We circumvent this issue by sampling from the loss function. But calculating the loss function, which in this case is based on some independence criteria is computationally expensive.

	$n = 4$	$n = 8$	$n = 16$
100	0.01	0.03	0.06
200	0.01	0.03	0.07
400	0.01	0.03	0.07
800	0.02	0.04	0.08
1600	0.02	0.05	0.12
3200	0.03	0.09	0.19

Table 5.1 Time taken to evaluate the SNR IC based loss function. Rows are the number of observations, columns are the number of nodes.

From Tables 5.1-5.3 we observe that time taken to evaluate the independence criterion based loss function grows approximately linearly with respect to the number of nodes n. This is expected as after updating an edge (and therefore updating the residuals of a node) we have to calculate the independence criterion value for all the pairs of nodes including the updated node. A more interesting observation is the increase in time with respect to the number of observations. The computational time for dCov

	$n = 4$	$n = 8$	$n = 16$
100	0.00	0.01	0.02
200	0.01	0.02	0.05
400	0.03	0.07	0.17
800	0.17	0.39	0.90
1600	0.93	2.36	4.89
3200	5.53	12.86	27.89

Table 5.2 Time taken to evaluate the dCov based loss function. Rows are the number of observations, columns are the number of nodes.

	$n = 4$	$n = 8$	$n = 16$
100	0.05	0.07	0.18
200	0.06	0.13	0.29
400	0.11	0.26	0.53
800	0.21	0.53	1.06
1600	0.53	1.14	2.37
3200	1.24	2.99	6.48

Table 5.3 Time taken to evaluate the HSIC based loss function. Rows are the number of observations, columns are the number of nodes.

increases approximately quadratic and the computational time for HSIC increases approximately linearly. This is expected as dCov requires multiplying matrices of order $n^2 \times n^2$ while HSIC uses incomplete Cholesky decomposition so needs to multiply matrices of order $n^2 \times k$, wehere k is the number of eigenvalues we keep. HSIC is slightly slower than dCov for the datasets with less than 800 observations, this can be explained by their implementation. dCov is implemented in `C++` and HSIC is currently only implemented in `R`, which is computationally much slower programming language. On the other hand for the datasets with more than 800 observations HSIC is faster. This can be explained as HSIC is using incomplete Cholesky decomposition while the dCov implementation in the Energy Rizzo and Szekely (2014) package does not use it, therefore for very high dimensional matrices, the slower `R` implementation with the incomplete Cholesky decomposition becomes more time efficient than the full dimension matrix multiplication using `C++` implementation. Finally the SNRIC shows only marginal increase in the computational time with the number of observations. This makes SNRIC more suitable for network inference when large number of observations is present.

Lets make a crude time estimate for inferring a network using the loss function based MCMC. Lets consider a graph on 16 nodes with 300 observations. From Table 2.3 we need at least 10^5 structure sampling iterations to get a satisfactory result. As in this case we are sampling parameters as well as structure, we have to run our algorithm for at least 2^6 iterations. For the network of this size we would need to run one iteration for 0.06 sec for SNR IC, 0.17 sec for dCov and 0.5 sec for HSIC based algorithm. To run the algorithm would approximately take between 33h for SNR to 294h for HSIC. This is a very crude estimate and this calculation could significantly benefit from running parallel MCMC. We can see that even a network of modest size can be very time consuming to infer.

5.8.2 Future Work

Future work could take two main directions.
First avenue is increasing the speed of the algorithm. As we have discussed in Section 4.5.2 the first step would be to implement SNR IC, HSIC and dCov with incomplete Cholesky decomposition in a fast compiled language such as `C++`. Next step would be to explore the fast versions of current independence criteria, for example fast version of dCov (Huo and Székely, 2016).
Other approach to continue this work would be to use a mutual independence criteria. Current implementation defines loss function as a sum of pair-wise independence criteria over all pairs of residuals. It is sufficient if we assume that data comes from the additive noise model. If we would choose to relax this assumption we would need to test for mutual independence rather than pairwise independence.

Conclusions

In this final chapter we summarise the key points considered in this thesis and place our contributions in the context of other work in the field. We finish off by describing some of the limitations of this work, and suggest possible areas for future research.

In Chapter 1 we introduced a way of combining graph theory and probability theory as a framework for probabilistic graphical models. We then use this framework for network inference; we represent a network with a graph and the underlying probability distribution.

In Chapter 2 we discussed the discrete Bayesian network inference using the MCMC sampler technique. The main goal was to introduce the structure search and discuss efficient MCMC sampling using parallel tempering. We applied this algorithm to simulated data as well as discretized experimental data. We discussed two approaches for datasets with missing values. Imputing values in preprocessing, for example using `R` package `MICE`, and a full MCMC sampler which samples the network structures as well as the missing values. Both approaches provided very similar results in terms of the network inference. Imputing the missing values in the preprocessing step is significantly faster therefore this approach is recommended when dealing with large networks, when a full sampler becomes too slow. Finally we showed that the likelihood equivalence non-preserving priors affect whether a variable with more uncertainty is more likely to be the causal one. For the possible future extensions we propose to consider expanding a possible MCMC jump space, for example inverting entire chain subgraph, resampling all the parents of the node, etc. It would require additional work in order to establish the jump probabilities for these more complex moves and their inverses.

In Chapter 3 we discussed three independence criteria from the literature. Namely, distance covariance criterion (dCov), kernel canonical correlation (KCC) and Hilbert-Schmidt Independence Criterion (HSIC). For each, we provided a simple geometrical interpretation which was not given in the original papers. We also established explicit relationships between them showing how each could be considered as a special case of HSIC. Both dCov and HSIC showed similar performance in term of testing for

independence. dCov is slightly faster to calculate therefore might be preferable to HSIC. We also introduced a new approach to estimating the dependence between variables from an additive and multiplicative noise model: the SNRIC. SNRIC is not as general as the previous three criteria, but it works in most of the practical cases and is significantly faster to calculate than both dCov and HSIC. SNRIC performed similarly to dCov and HSIC in testing for independence. SNRIC performed slightly better then dCov and HSIC when dealing with heteroscedastic data and data where dependence is due to latent variables and performed slightly worse on data with periodic relationships. An interesting avenue for the future work would be to investigate the effect of other kernels on HSIC, for example linear, polynomial, hyperbolic tangent, Laplacian or Bessel. They are conveniently implemented in `kernlab` (Karatzoglou and Smola, 2003) package for the `R` statistical environment (R Core Team, 2014). Current implementation of SNRIC uses polynomial regression. This choice was made to maximize the computational efficiency while preserving sufficiently good performance. Future extensions of the method would involve finding a better regression method that would provide better accuracy for the method while not compromising its computational efficiency. Possible approach would be to use splines (for example cubic splines, B-splines or P-splines).
In Chapter 4 we discussed a greedy search based PC algorithm extension: Independence Criterion based PC or ICPC. ICPC uses a general independence criterion rather than a Fisher's z-test. ICPC using HSIC (kernel PC or kPC) was introduced in Gretton et al. (2009) and Tillman (2009). We developed an alternative to kPC: dPC using the distance covariance criterion. The kPC algorithm in the original paper only used a full conditional independence test (cluster permutation test). We developed a faster approach to testing for conditional independence using unconditional independence test on the residuals after regressing variables on the conditional set. To the best of our knowledge the kPC algorithm has not yet been implemented in a ready to use package. We implemented kPC and dPC algorithms in an `R` package `kpcalg` (Verbyla et al., 2017). All algorithms performed similarly in terms of finding the underlying network. kPC-cluster which uses cluster permutation test is significantly slower than the kPC-Resid and the dPC-Resid. Due to similar performance but different implementation we suggest to use dPC-Resid for networks with less than 1000 observations and kPC-Resid for networks with more than 1000 observations. The first avenue of future work is to implement the general independence criteria based tests in `C++` or other compiled programming language. This would greatly increase the computational time of the algorithms. Another avenue for the future work is more theoretical rather than programming based. Currently we are only aware of

the Cluster-HSIC conditional independence test which is significantly slower than the residual counterparts. ICPC method would strongly benefit if we could develop a fast full conditional independence test and use it instead of either the residuals based independence tests or the Cluster-HSIC.

We started by defining the framework in Chapter 1, we introduced the MCMC techniques for network inference in Chapter 2, we discuss the general independence criteria in Chapter 3 we showed the potential of using independence criteria in network inference in Chapter 4 and finally we combine everything in one algorithm in Chapter 5.

In Chapter 5 we presented an MCMC sampler using a general loss function. Doing so allowed us to sample from any loss function instead of the likelihood. We used a loss function based on the independence criterion. This allowed us to sample from the distribution defined by the amount of dependence left in the residuals. This is a very flexible approach because it does not assume a functional form of the relationship between variables or the noise distributions and allows cyclic relationships between variables.

It is important to note that the MCMC sampler from loss function algorithm presented in Chapter 5 still has some shortcomings. In the experimental datasets it is often the case that latent variables are present. Current iteration of the algorithm is unable to deal with latent variables. A possible future development would be to incorporate latent variables into the MCMC sampler. Another issue is missing data, we used `R` package `MICE` to impute the missing data. A full MCMC sampler that samples network structure and parameters, and imputes the missing data (similar to the MCMC sampler in Chapter 2) would be another natural extension. We need to choose an independence criterion for the loss function. SNRIC is fast to calculate, but in general, it is an approximation of a general independence criterion. In Chapter 5 we assume that the data comes from an additive noise model so SNRIC is an exact independence criterion rather than just an approximation of a general independence criterion. Although if we would like to relax the additive noise model condition, SNRIC might not work any more. HSIC and dCov are general independence criteria, but both are expensive to calculate. Therefore it would be convenient to have a fast implementation of a general independence criterion. Finally, although we used a pairwise independence criterion for the loss function, it would be desirable to use a mutual independence criterion in the future.

References

Altay, G. and Emmert-Streib, F. (2010). Revealing differences in gene network inference algorithms on the network level by ensemble methods. *Bioinformatics 26 (14) (2010) 1738–1744*.

Aluru, S. (2006). *Handbook of Computational Molecular Biology*. Chapman and Hall/CRC, Boca Raton, FL.

Antti Hyttinen, Frederick Eberhardt, P. O. H. (2012). Learning linear cyclic causal models with latent variables. *Journal of Machine Learning Research 13(Nov):3387-3439, 2012*.

Bach, F. R. and Jordan, M. I. (2002). Kernel independent component analysis. *The Journal of Machine Learning Research 3(Jul):1-48, 2002*, 3:1–48.

Banf, M. and Rhee, S. Y. (2017). Computational inference of gene regulatory networks: Approaches, limitations and opportunities. *Biochim Biophys Acta. 2017 Jan;1860(1):41-52*.

Bansal, M., Gatta, G., and di Bernardo, D. (2006). Inference of gene regulatory networks and compound mode of action from time course gene expression profiles. *Bioinformatics, 22 (7) (2006)*, pages 815–822.

Basso, K., Margolin, A., Stolovitzky, G., Klein, U., Dalla-Favera, R., and Califano, A. (2005). Reverse engineering of regulatory networks in human b cells. *Nat. Genet., 37 (2005)*, pages 382–390.

Berg, C., Christensen, J. P. R., and Ressel, P. (1984). *Harmonic Analysis on Semigroups*. Springer, New York.

Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer, New York.

Bissiri, P., Holmes, C., and Walker, S. G. (2016). A general framework for updating belief distributions. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, Volume 78(Issue 5, November 2016):1103–1130.

Breiman, L. (2001). Random forests. *Mach. Learn. 45 (1) (2001) 5–32*.

Brooks, S. P., Giudici, P., and Roberts, G. (2003). Efficient construction of reversible jump markov chain monte carlo proposal distributions. *Journal of the Royal Statistical Society: Series B (Statistical Methodology) 65, 3–39*.

Butte, A. and Kohane, I. (2000). Mutual information relevance networks: functional genomic clustering using pairwise entropy measurements. *Proceeding of the Pacific Symposium on Biocomputing (2000)*, pages 418–429.

de Boor, C. (1978). *A Practical Guide to Splines*. Springer-Verlag.

Denker, M. (1985). *Asymptotic Distribution Theory in Nonparametric Statistics*. Fr. Vieweg & Sohn, Braunschweig, Wiesbaden.

di Bernardo, D., Thompson, M., Gardner, T., Chobot, S., Eastwood, E., Wojtovich, A., Elliott, S., Schaus, S., and Collins, J. (2005). Chemogenomic profiling on a genome-wide scale using reverse-engineered gene networks. *Nat. Biotechnol., 23 (3) (2005)*, pages 377–383.

Dunne, D. W., Butterworth, A. E., Fulford, A. J., Kariuki, H., Langley, J., Ouma, J., Capron, A., Pierce, R. J., and Sturrock, R. F. (1992). Immunity after treatment of human schistosomiasis: association between ige antibodies to adult worm antigens and resistance to reinfection. *European J. Immunology, 22(6), 1483–94*.

Earl, D. and Deem, W. (2008). Parallel tempering: Theory, applications, and new perspectives. *Physical Chemistry Chemical Physics Issue 23, 2005: 3910-6.*

Eilers, P. H. C. and Marx, B. D. (1996). Flexible smoothing with b-splines and penalties. *Statist. Sci. Volume 11, Number 2 (1996), 89-121.*

Faith, J., Hayete, B., Thaden, J., Mogno, I., Wierzbowski, J., Cottarel, G., Kasif, S., Collins, J., and Gardner, T. (2007). Large-scale mapping and validation of escherichia coli transcriptional regulation from a compendium of expression profiles. *PLoS Biol., 5 (1) (2007), p. e8.*

Ferguson, T. S. (2005). U-statistics. Notes for Statistics 200C, Spring 2005.

Filkov, V. (2005). *Handbook of computational molecular biology*. Chapman & Hall/CRC Computer and Information Science Series.

Fisher, F. (1970). A correspondence principle for simultaneous equation models. *Econometrica, 38(1), 73-92. doi:10.2307/1909242.*

Fukumizu, K., Gretton, A., Sun, X., and Schölkopf, B. (2008). Kernel measures of conditional dependence. In Platt, J. C., Koller, D., Singer, Y., and Roweis, S. T., editors, *Advances in Neural Information Processing Systems 20*, pages 489–496. Curran Associates, Inc.

Gardner, T., di Bernardo, D., Lorenz, D., and Collins, J. (2003). Inferring genetic networks and identifying compound mode of action via expression profiling. *Science, 301 (2003)*, pages 102–105.

Geman, S. and Geman, D. (1984). Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Trans. Pattern Anal. Mach. Intell.*, 6(6):721–741.

Green, P. J. (1995). Reversible jump markov chain monte carlo computation and bayesian model determination. *Biometrika, 82 (4): 711–732.*

Gretton, A., Bousquet, O., Smola, A., and Schölkopf, B. (2005). Measuring statistical dependence with hilbert-schmidt norms. In *Algorithmic learning theory*, pages 63–77. Springer.

Gretton, A., Fukumizu, K., Teo, C. H., Song, L., Schölkopf, B., and Smola, A. J. (2008). A kernel statistical test of independence. In Platt, J. C., Koller, D., Singer, Y., and Roweis, S. T., editors, *Advances in Neural Information Processing Systems 20*, pages 585–592. Curran Associates, Inc.

Gretton, A., Spirtes, P., and Tillman, R. E. (2009). Nonlinear directed acyclic structure learning with weakly additive noise models. In Bengio, Y., Schuurmans, D., Lafferty, J. D., Williams, C. K. I., and Culotta, A., editors, *Advances in Neural Information Processing Systems 22*, pages 1847–1855. Curran Associates, Inc.

Hamze, F., Dickson, N., and Karimi, K. (2010). Robust parameter selection for parallel tempering. *International Journal of Modern Physics C.*

Hartigan, J. A. and Wong, M. A. (1979). A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics) Vol. 28, No. 1 (1979)*, 28:100–108.

Hastie, T. and Tibshirani, R. (1986). Generalized additive models. *Statistical Science, Vol. 1, No. 3 (Aug., 1986)*, pages 297–310.

Hastings, W. K. (1970). Monte carlo sampling methods using markov chain and their applications. *Biometrika (1970), 57, 1, p. 97.*

Haury, A.-C., Mordelet, F., Vera-Licona, P., and Vert, J.-P. (2012). Tigress: Trustful inference of gene regulation using stability selection. *BMC Syst. Biol. 6 (2012) 145.*

Hecker, M., Lambeck, S., Toepfer, S., van Someren, E., and Guthke, R. (2009). Gene regulatory network inference: Data integration in dynamic models—a review. *Biosystems*, Volume 96(Issue 1, April 2009):86–103.

Heckerman, D. (1996). A tutorial on learning with bayesian networks. *Microsoft Research Tech. Report, MSR-TR-95-06.*

Heckerman, D., Meek, C., and Cooper, G. (1997). A bayesian approach to causal discovery. Technical report, Microsoft Research.

Hoyer, P. O., Janzing, D., Mooij, J. M., Peters, J., and Schölkopf, B. (2009). Nonlinear causal discovery with additive noise models. In Koller, D., Schuurmans, D., Bengio, Y., and Bottou, L., editors, *Advances in Neural Information Processing Systems 21*, pages 689–696. Curran Associates, Inc.

Hunter, J. K. and Nachtergaele, B. (2001). *Applied Analysis.* World Scientific Pub. Co. Inc.

Huo, X. and Székely, G. J. (2016). Fast computing for distance covariance. *Technometrics*, 58(4):435–447.

Huynh-Thu, V., Irrthum, A., Wehenkel, L., and Geurts, P. (2010). Inferring regulatory networks from expression data using tree-based methods. *PloS one 5 (9). (2010)*, 5(9):e12776.

Hyttinen, A., Eberhardt, F., and Hoyer, P. O. (2010). Causal discovery for linear cyclic models with latent variables. *on Probabilistic Graphical Models*, page 153.

Iba, Y. (2001). Extended ensemble monte carlo. *Int.J.Mod.Phys. C12 (2001) 623-656.*

Ihaka, R. and Gentleman, R. (1996). R: A language for data analysis and graphics. *J. of Comp. and Graphical Statistics Vol. 5, No. 3 (Sep., 1996)*, pages 299–314.

Karatzoglou, A. and Smola, A. (2003). kernlab – a kernel methods package. *Proceedings of the 3rd International Workshop on Distributed Statistical Computing (DSC 20013).*

Koller, D. and Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques*. MIT Press.

Lacerda, G., Spirtes, P., Ramsey, J., and Hoyer, P. O. (2008). Discovering cyclic causal models by independent components analysis. *Proceedings of the Twenty-Fourth Conference on Uncertainty in Artificial Intelligence (UAI2008).*

Lee, A. J. (1990). *U-Statistics*. New York : Marcel Dekker.

Liang, S., Fuhrman, S., and Somogyi, R. (1998). Reveal, a general reverse engineering algorithm for inference of genetic network architectures. *Proceeding of the Pacific Symposium on Biocomputing (1998)*, pages 18–29.

Lippert, C., Stegle, O., Ghahramani, Z., and Borgwardt, K. (2009). A kernel method for unsupervised structured network inference. *JMLR Workshop and Conference Proceedings Volume 5:368-375.*

Luo, W., Hankenson, K., and Woolf, P. (2008). Learning transcriptional regulatory networks from high throughput gene expression data using continuous three-way mutual information. *BMC Bioinf. 9 (2008) 467.*

MacKay, D. (2003). *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press.

Mangan, S. and Alon, U. (2003). Structure and function of the feed-forward loop network motif. *Proc. Natl. Acad. Sci. U. S. A. 100 (21) (2003) 11980–11985.*

Marbach, D., Costello, J. C., Küffner, R., Vega, N. M., Prill, R. J., Camacho, D. M., amd The DREAM5 Consortium, K. R. A., Kellis, M., Collins, J. J., and Stolovitzky, G. (2012). Wisdom of crowds for robust gene network inference. *Nature Methods 9, 796–804 (2012).*

Marbach, D., Prill, R. J., Schaffter, T., Mattiussi, C., Floreano, D., and Stolovitzky, G. (2010). Revealing strengths and weaknesses of methods for gene network inference. *Proc Natl Acad Sci USA 2010; 107:6286–6291.*

Margolin, A., Nemenman, I., Basso, K., Wiggins, C., Stolovitzky, G., and R. Favera, A. C. (2006). Aracne: an algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context. *BMC Bioinform., 7 (Suppl. 1) (2006), p. S7.*

Meek, C. (1995). Causal inference and causal explanation with background knowledge. In *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, pages 403–410. Morgan Kaufmann Publishers Inc.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953). Equation of state calculations by fast computing machines. *J. Chem. Phys. 21, 1087 (1953)*, 21(6):1087–1092.

Meyer, P. (2010). Information-theoretic inference of gene networks using backward elimination. *Conference on bioinformatics and computational biology, 2010,.*

Meyer, P. E., Kontos, K., Lafitte, F., and Bontempi, G. (2007). Information-theoretic inference of large transcriptional regulatory networks. *EURASIP J Bioinform Syst Biol. 2007:79879*, 2007(1):1–9.

Murphy, K. P. (2012). *Machine Learning A Probabilistic Perspective*. The MIT Press.

Needham, C., Bradford, J., Bulpitt, A., and Westhead, D. (2007). A primer on learning in bayesian networks for computational biology. *PLoS Comput. Biol., 3 (8) (2007), p. e129.*

Noor, A., Serpedin, E., Nounou, M., Nounou, H., Mohamed, N., and Chouchane, L. (2013). An overview of the statistical methods used for inferring gene regulatory networks and protein-protein interaction networks. *Advances in Bioinformatics, 2013, 953814.*

Obayashi, T. and Kinoshita, K. (2009). Rank of correlation coefficient as a comparable measure for biological significance of gene coexpression. *DNA research 2009*, 16(5):249–260.

Opgen-Rhein, R. and Strimmer, K. (2007). From correlation to causation networks: a simple approximate learning algorithm and its application to high-dimensional plant gene expression data. *BMC systems biology 2007; 1: 37.*, 1(1):37.

Pearl, J. (2000). *Causality: models, reasoning and inference*, volume 29. Cambridge Univ Press.

Perrin, B., Ralaivola, L., Mazurie, A., Bottani, S., Mallet, J., and d'Alché Buc, F. (2003). Gene networks inference using dynamic bayesian networks. *Bioinformatics 19 (Suppl. 2), ii138–ii148.*

R Core Team (2014). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Rao, A., Hero III, A., States, D., and Engel, J. (2007). Using directed information to build biologically relevant influence networks. *Comput. Syst. Bioinformatics Conf. 6, 145–156.*

Richardson, T. S. (1996). A discovery algorithm for directed cyclis graphs. *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence (UAI1996)*.

Rizzo, M. L. and Szekely, G. J. (2014). *E-statistics*.

Sachs, K., Perez, O., Pe'er, D., Lauffenburger, D. A., and Nolan, G. P. (2005). Causal protein-signaling networks derived from multiparameter single-cell data. *Science 2005 Aug 19;309(5738):1187*, 308(5721):523–529.

Savageau, M. (1970). *Biochemical Systems Analysis*. Addison-Wesley, Reading.

Schölkopf, B. (2001). The kernel trick for distances. In Leen, T. K., Dietterich, T. G., and Tresp, V., editors, *Advances in Neural Information Processing Systems 13*, pages 301–307. MIT Press.

Sejdinovic, D., Sriperumbudur, B., Gretton, A., Fukumizu, K., et al. (2013). Equivalence of distance-based and rkhs-based statistics in hypothesis testing. *The Annals of Statistics 2013*, 41(5):2263–2291.

Shimizu, S., Hoyer, P. O., Hyvärinen, A., and Kerminen, A. (2006). A linear non-gaussian acyclic model for causal discovery. *Journal of Machine Learning Research*, 7(Oct):2003–2030.

Soranzo, N., Bianconi, G., and Altafini, C. (2007). Comparing association network algorithms for reverse engineering of large-scale gene regulatory networks: synthetic versus real data. *Bioinformatics 23 (13) (2007) 1640–1647*.

Spirtes, P. and Glymour, C. N. (1990). A fast algorithm for discovering sparse causal graphs. Technical report, Carnegie Mellon University.

Steuer, R., Kurths, J., Daub, C., Weise, J., and Selbig, J. (2002). The mutual information: detecting and evaluating dependencies between variables. *Bioinformatics, 18 (Suppl. 2) (2002)*, pages S231–S240.

Stuart, J., Segal, E., Koller, D., and Kim, S. (2003). A gene-coexpression network for global discovery of conserved genetic modules. *Science, 302 (5643) (2003)*, pages 249–255.

Sutherland, W. A. (2009). *Introduction to Metric and Topological Spaces (2nd Edition)*. Oxford University Press.

Székely, G. J., Rizzo, M. L., Bakirov, N. K., et al. (2007). Measuring and testing dependence by correlation of distances. *The Annals of Statistics Volume 35, Number 6 (2007), 2769-2794*, 35(6):2769–2794.

Székely, G. J., Rizzo, M. L., et al. (2009). Brownian distance covariance. *The Annals of Applied Statistics Volume 3, Number 4 (2009), 1236-1265.*, 3(4):1236–1265.

Tasaki, S., Sauerwine, B., Hoff, B., Toyoshiba, H., Gaiteri, C., and Neto, E. C. (2015). Bayesian network reconstruction using systems genetics data: comparison of mcmc methods. *Genetics 199 (4) (2015) 973–989*.

The Carter Center (2014). Schistosomiasis control program.

Thétiot-Laurent, S., Boissier, J., Robert, A., and Meunier, B. (2013). Schistosomiasis chemotherapy. *Angew Chem Int Ed Engl. 2013 Jul 29;52(31):7936-56. doi: 10.1002/anie.201208390.*

Tillman, R. E. (2009). Learning directed graphical models from nonlinear and non-gaussian data. *Master thesis.*

Tukahebwa, E. M., Magnussen, P., Madsen, H., Kabatereine, N. B., Nuwaha, F., Wilson, S., and Vennervald, B. J. (2013). A very high infection intensity of schistosoma mansoni in a ugandan lake victoria fishing community is required for association with highly prevalent organ related morbidity. *PLoS Negl Trop Dis. 2013 Jul 25;7(7):e2268. doi: 10.1371/journal.pntd.0002268. Print 2013.*

Usadel, B., Obayashi, T., Mutwil, M., Giorgi, F., Bassel, G., Tanimoto, M., Chow, A., Steinhauser, D., Persson, S., and Provart, N. (2009). Co-expression tools for plant biology: opportunities for hypothesis generation and caveats. *Plant Cell Environ. 32 (12) (2009) 1633–1651.*

Van Berlo, R., van Someren, E., and Reinders, M. (2003). Studying the conditions for learning dynamic bayesian networks to discover genetic regulatory networks. *Simul.: Trans. Soc. Model. Simul. Int. 79 (12), 689–702.*

van Buuren, S. and Oudshoorn, K. (1999). Flexible multivariate imputation by mice. Technical report, TNO report PG/VGZ/99.054.

Verbyla, P., Desgranges, N. I. B., and Wernisch, L. (2017). *kpcalg: Kernel PC Algorithm for Causal Structure Detection.*

Wald, A. (1950). *Statistical decision functions.* Wiley.

Wang, Z. and de Freitas, N. (2011). Predictive adaptation of hybrid monte carlo with bayesian parametric bandits. *NIPS24 Deep Learning and Unsupervised Feature Learning Workshop.*

Wernisch, L., Houghton, J., and D., D. (2007). Structural equation modeling of immune response to schistosomiasis infection. Technical report, MRC Biostatistic Unit.

Whittaker, E. and Watson, G. (1996). *A course of modern analysis.* Cambridge University Press.

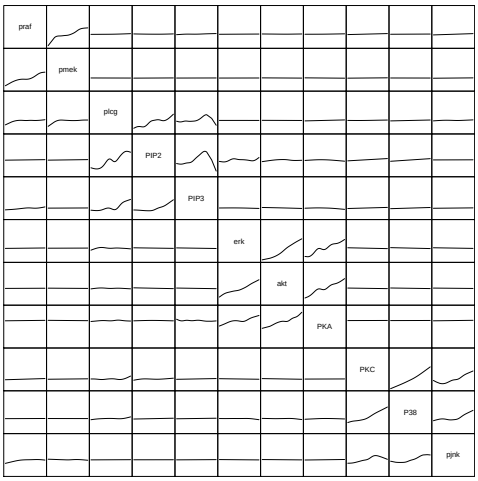
Wood, S. (2004). Stable and efficient multiple smoothing parameter estimation for generalized additive models. *Journal of the American Statistical Association*, 99:673–686.

Wood, S. (2011). Fast stable restricted maximum likelihood and marginal likelihood estimation of semiparametric generalized linear models. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 73(1):3–36.

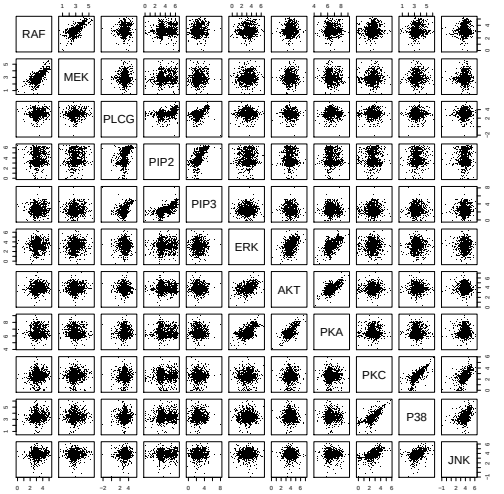
Zhang, B. and Horvath, S. (2005). A general framework for weighted gene co-expression network analysis. *Stat. Appl. Genet. Mol. Biol. 4 (2005) Article17.*

Appendix A

Data



(a) Signals between proteins in the simulated dataset 8.



(b) Plot of the data in the simulated dataset 8.

Fig. A.1 Data simulated from the single-cell dataset 8 after the log - transformation (from Sachs et al. (2005)).

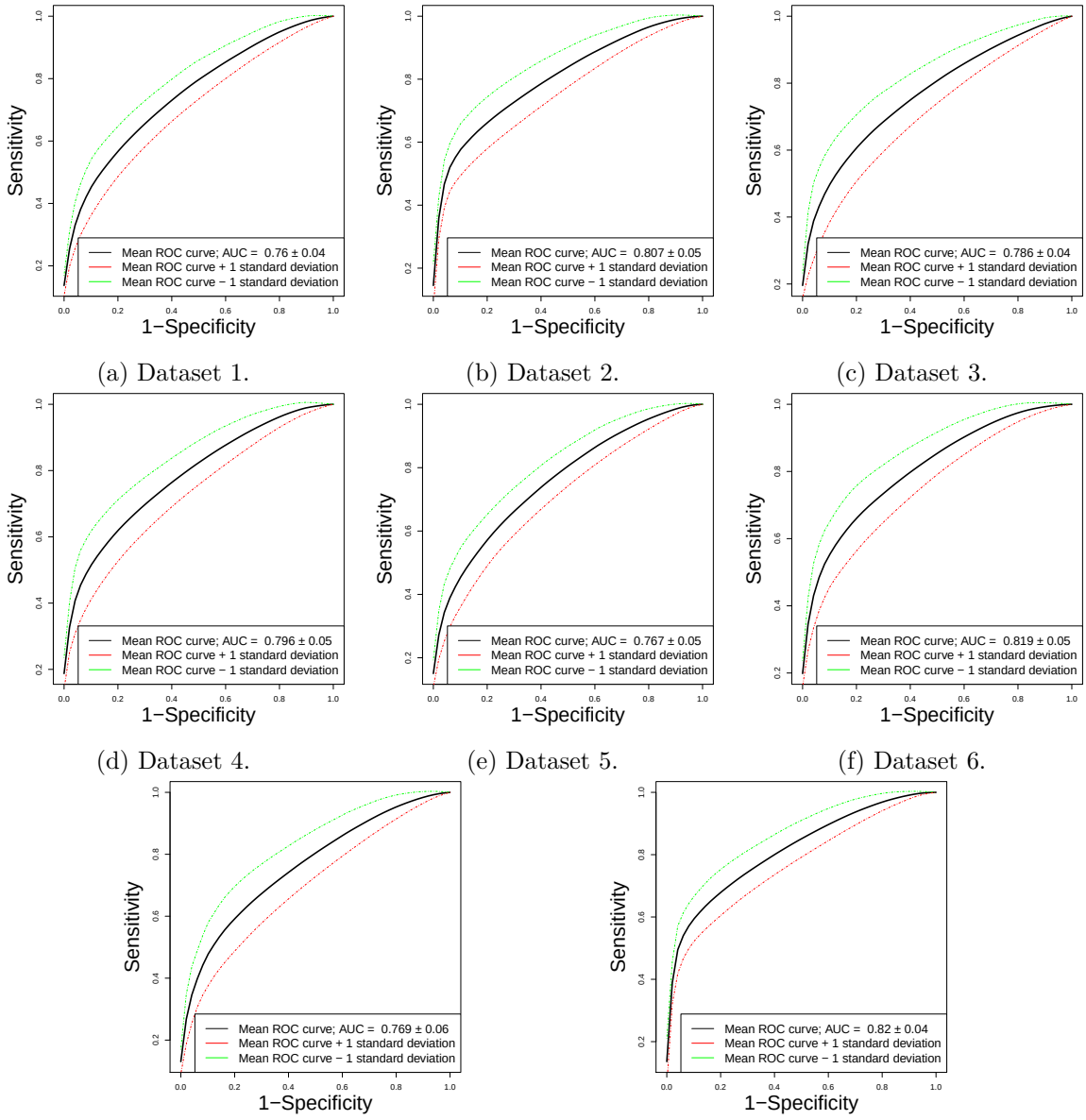
Name	Type	Time	Missing values
Sex	Demographic	Before treatment	0
Age	Demographic	Before treatment	0
Weight	Demographic	Before treatment	0
Egg count pre	Infection load	Before treatment	0
IL-4 pre	Interleukin	Before treatment	136
IL-5 pre	Interleukin	Before treatment	141

IL-10 pre	Interleukin	Before treatment	44
IL-13 pre	Interleukin	Before treatment	134
IL-4R pre	Interleukin	Before treatment	215
IL-5R pre	Interleukin	Before treatment	137
IL-13R pre	Interleukin	Before treatment	135
Eutaxin pre	Immunoglogulin	Before treatment	80
IgE pre	Immunoglogulin	Before treatment	24
IgG1 pre	Immunoglogulin	Before treatment	12
IgG4 pre	Immunoglogulin	Before treatment	11
IL-5 24 hrs	Interleukin	24h after treatment	134
IL-10 24 hrs	Interleukin	24h after treatment	62
IL-13 24 hrs	Interleukin	24h after treatment	164
IL-4R 24 hrs	Interleukin	24h after treatment	187
IL-5R 24 hrs	Interleukin	24h after treatment	119
IL-13R 24 hrs	Interleukin	24h after treatment	126
Eutaxin 9 weeks	Immunoglogulin	9 weeks after treatment	125
IgE 9 weeks	Immunoglogulin	9 weeks after treatment	29
IgG1 9 weeks	Immunoglogulin	9 weeks after treatment	57
IgG4 9 weeks	Immunoglogulin	9 weeks after treatment	26
Egg count 8 months	Infection load	8 months after treatment	28
Egg count 2 years	Infection load	2 years after treatment	53

Table A.1 Schistosomiasis dataset; all variables

Appendix B

Discrete Bayesian Network



(a) Dataset 1. (b) Dataset 2. (c) Dataset 3.

(d) Dataset 4. (e) Dataset 5. (f) Dataset 6.

(g) Dataset 7. (h) Dataset 8.

Fig. B.1 Discrete MCMC effectiveness on data simulated from the single-cell data, all eight datasets.

Appendix C

Independence Criteria

C.1 $\mathcal{F}$-correlation

Proposition C.1.1. Let $(x, y) = ((x_1, y_1), ..., (x_n, y_n))$ be observations from the two random variables X and Y. Let K and L, defined as $K_{ij} = k(x_i, x_j)$ and $L_{ij} = l(y_i, y_j)$ for some kernels k and l, be their kernel matrices respectively, then

$$\max_{(f,g)\in\mathcal{F}_\phi\times\mathcal{G}_\psi} \widehat{\mathrm{Cov}}(f(x)g(y)) = \max_{\alpha,\beta\in\mathbb{R}^n} \frac{1}{n}\alpha^T\tilde{K}\tilde{L}\beta \tag{C.1.1}$$

where $\tilde{K} = KH$, $\tilde{L} = LH$ and H is a matrix with elements $H_{ij} = \delta_{ij} - 1/n$.

Proof.

$$
\begin{aligned}
\max_{(f,g)\in\mathcal{F}_\phi\times\mathcal{G}_\psi} \widehat{\mathrm{Cov}}(f(X)g(Y)) &= \max_{(f,g)\in\mathcal{F}_\phi\times\mathcal{G}_\psi} \hat{\mathrm{E}}_{X,Y}\Big[\Big(f(X)-\hat{\mathrm{E}}_X[f(X)]\Big)\Big(g(Y)-\hat{\mathrm{E}}_Y[g(Y)]\Big)\Big] \\
&= \max_{(f,g)\in\mathcal{F}_\phi\times\mathcal{G}_\psi} \hat{\mathrm{E}}_{X,Y}\Big[f(X)g(Y)\Big] - \hat{\mathrm{E}}_X\Big[f(X)\hat{\mathrm{E}}_Y\Big[g(Y)\Big]\Big] \\
&\quad - \hat{\mathrm{E}}_Y\Big[\hat{\mathrm{E}}_X\Big[f(X)\Big]g(Y)\Big] + \hat{\mathrm{E}}_X\Big[f(X)\Big]\hat{\mathrm{E}}_Y\Big[g(Y)\Big] \\
&= \max_{\alpha,\beta\in\mathbb{R}^n} \frac{1}{n}\sum_{l=1}^n \underbrace{\Big[\sum_{i=1}^n \alpha_i k(x_i,x_l)\Big]}_{f(x_l)} \underbrace{\Big[\sum_{j=1}^n \beta_j k(y_j,y_l)\Big]}_{g(y_l)} \\
&\quad - \frac{1}{n}\sum_{l=1}^n \underbrace{\Big[\frac{1}{n}\sum_{k=1}^n\sum_{i=1}^n \alpha_i k(x_i,x_k)\Big]}_{\hat{\mathrm{E}}_X[f(X)]} \Big[\sum_{j=1}^n \beta_j k(y_j,x_l)\Big] \\
&\quad - \frac{1}{n}\sum_{l=1}^n \Big[\sum_{i=1}^n \alpha_i k(x_i,x_l)\Big] \underbrace{\Big[\frac{1}{m}\sum_{k=1}^n\sum_{j=1}^n \beta_j k(y_j,y_k)\Big]}_{\hat{\mathrm{E}}_Y[g(Y)]} \\
&\quad + \Big[\frac{1}{n}\sum_{l=1}^n \Big(\sum_{i=1}^n \alpha_i k(x_i,x_l)\Big)\Big]\Big[\frac{1}{m}\sum_{k=1}^n \Big(\sum_{j=1}^n \alpha_i k(y_j,y_k)\Big)\Big] \\
&= \max_{\alpha,\beta\in\mathbb{R}^n} \frac{1}{n}\alpha^T KL\beta - \frac{1}{m}\Big[(\frac{1}{n}\alpha^T K\mathbf{1})\mathbf{1}^T\Big]\Big[L\beta\Big] \\
&\quad - \frac{1}{n}\Big[\alpha^T K\Big]\Big[\mathbf{1}(\frac{1}{n}\mathbf{1}^T L\beta)\Big] + \Big[(\frac{1}{n}\alpha^T K\mathbf{1})\mathbf{1}^T\Big]\Big[\mathbf{1}(\frac{1}{n}\mathbf{1}^T L\beta)\Big] \\
&= \max_{\alpha,\beta\in\mathbb{R}^n} \frac{1}{n}\alpha^T (K-\frac{1}{m}K\mathbf{1}\mathbf{1}^T)(L-\frac{1}{m}L\mathbf{1}\mathbf{1}^T)\beta \\
&= \max_{\alpha,\beta\in\mathbb{R}^n} \frac{1}{n}\alpha^T K(I-\frac{1}{m}\mathbf{1}\mathbf{1}^T)L(I-\frac{1}{m}\mathbf{1}\mathbf{1}^T)\beta \\
&= \max_{\alpha,\beta\in\mathbb{R}^n} \frac{1}{n}\alpha^T KHLH\beta
\end{aligned}
$$

□

C.1.1 Regularization of the $\mathcal{F}$-correlation

Bach and Jordan (2002) points out that the un-regularised $\mathcal{F}$-correlation is not a particularly useful estimate. In many cases (for example for Gaussian kernels and distinct data points) the kernel matrices are invertible and the $\mathcal{F}$-correlation is identically equal to one.

Proposition C.1.2. Let K be an $n \times n$ invertible matrix, then the rows of K span $\mathbb{R}^n$.

Proof. Let $(e_1, ..., e_n)$ be the usual basis for $\mathbb{R}^n$, i.e. $(e_i)_j = \delta_{ij}$. Then any vector $v \in \mathbb{R}^n$ can be written as

$$v = \sum_{i=1}^{n} a_i e_i$$

for some constants a_i. As K is invertible, $K^{-1}v$ exists and again we can write it as

$$K^{-1}v = \sum_{i=1}^{n} b_i e_i$$

For some constants b_i. Then pre-multiplying by K yields

$$v = KK^{-1}v = \sum_{i=1}^{n} b_i (Ke_i)$$

We conclude that any vector $v \in \mathbb{R}^n$ can be written in terms of Ke_i, that is the rows of K. □

Denote the linear span of $\tilde{K}$ by $V_K \subset \mathbb{R}^n$ and the linear span of $\tilde{L}$ by $V_L \subset \mathbb{R}^n$. Consider the geometrical interpretation of Equation 3.3.14

$$\hat{\rho}_{\mathcal{F}}(x, y) = \max_{u \in V_K, v \in V_L} \frac{u^T v}{(u^T u)^{1/2} (v^T v)^{1/2}} = \cos(u, v)$$

The linear span of H (recall $H_{ij} = \delta_{ij} - 1/n$) is the orthogonal compliment of the vector of all ones. Then if K and L are invertible their linear spans are all of $\mathbb{R}^n$, and so the linear spans of $\tilde{K}$ and $\tilde{L}$ are the same. Therefore we can always take u equals v in the equation above which yields the empirical estimate of $\mathcal{F}$-correlation being identically one.

To overcome this issue Bach and Jordan (2002) introduced the *Regularised $\mathcal{F}$-correlation*:

$$\rho_{\mathcal{F}}^{\kappa} = \max_{(f,g) \in \mathcal{F} \times \mathcal{G}} \frac{\operatorname{Cov}(f(x)g(y))}{(\operatorname{Var}(f(x)) + \kappa \|f\|_{\mathcal{F}}^2)^{1/2} \left(\operatorname{Var}(g(y)) + \kappa \|g\|_{\mathcal{G}}^2\right)^{1/2}}$$

In order to find the empirical estimate of the regularised $\mathcal{F}$-correlation from a finite sample we expand $\operatorname{Var}(f(x)) + \kappa \|f\|_{\mathcal{F}}^2$ with respect to κ to obtain:

$$\begin{aligned}\operatorname{Var}(f(x)) + \kappa\|f\|_{\mathcal{F}}^2 &= \frac{1}{n}\alpha^T\tilde{K}^2\alpha + \kappa\alpha^T\tilde{K}\alpha \\ &= \frac{1}{n}\alpha^T\left(\tilde{K}^2 + n\kappa\tilde{K} + \frac{n^2\kappa^2}{4}I\right)\alpha - \alpha^T\alpha\frac{N\kappa^2}{4} \\ &\approx \frac{1}{n}\alpha^T\left(\tilde{K} + \frac{n\kappa}{2}I\right)^2\alpha, \qquad \text{for } \kappa \ll 1.\end{aligned}$$

This gives us the *Empirical Estimate* of Regularised $\mathcal{F}$-correlation $\rho_{\mathcal{F}}^{\kappa}$ as

$$\hat{\rho}_{\mathcal{F}}^{\kappa} = \max_{\alpha,\beta\in\mathbb{R}^n} \frac{\alpha^T\tilde{K}\tilde{L}\beta}{\left(\alpha^T\left(\tilde{K} + \frac{n\kappa}{2}I\right)^2\alpha\right)\left(\beta^T\left(\tilde{L} + \frac{n\kappa}{2}I\right)^2\beta\right)}$$

After using the regularization the generalized eigenvalue problem 3.3.20 becomes:

$$\begin{pmatrix} \left(\tilde{K} + \frac{n\kappa}{2}I\right)^2 & \tilde{K}\tilde{L} \\ \tilde{L}\tilde{K} & \left(\tilde{L} + \frac{n\kappa}{2}I\right)^2 \end{pmatrix}\begin{pmatrix} \alpha \\ \beta \end{pmatrix} = (1+\lambda)\begin{pmatrix} \left(\tilde{K} + \frac{n\kappa}{2}I\right)^2 & 0 \\ 0 & \left(\tilde{L} + \frac{n\kappa}{2}I\right)^2 \end{pmatrix}\begin{pmatrix} \alpha \\ \beta \end{pmatrix} \tag{C.1.2}$$

As before to find the $\mathcal{F}$-correlation we need to find the largest eigenvalue λ_{max} of C.1.2.

C.1.2 Kernel Generalized Variance

Proposition C.1.3. Two random variables X and Y are independent if and only if their mutual information $I(X,Y)$ is equal to zero.

Proof. "⇒" If X and Y are independent, then $p(x,y) = p(x)p(y)$. So $\log\frac{p(x,y)}{p(x)p(y)} = \log 1 = 0$, i.e. $I(X,Y) = 0$.

"$\Leftarrow$"

$$
\begin{aligned}
I(X,Y) &= \int_X \int_Y p(x,y) \log \frac{p(x,y)}{p(x)p(y)} dxdy \\
&= -\int_X \int_Y p(x,y) \log \frac{p(x)p(y)}{p(x,y)} \\
&= -\mathrm{E}_{X,Y}\left[\log \frac{p(x)p(y)}{p(x,y)} \right] \\
&\geq -\log\left(\mathrm{E}_{X,Y}\left[\frac{p(x)p(y)}{p(x,y)} \right] \right) && \text{by Jensen's inequality} \\
&= -\log\left(\int_{X,Y} p(x,y) \frac{p(x)p(y)}{p(x,y)} dxdy \right) \\
&= -\log\left(\int_{X,Y} p(x)p(y) dxdy \right) \\
&= -\log\left(\left(\int_X p(x)dx \right) \left(\int_Y p(y)dy \right) \right) \\
&= -\log 1 = 0
\end{aligned}
$$

with equality iff $\frac{p(x)p(y)}{p(x,y)}$ is constant, i.e. $p(x,y) = \alpha p(x)p(y)$. But this can only happen if $p(x,y) = p(x)p(y)$, i.e. X and Y are independent. □

Proposition C.1.4. Let $X \subset \mathbb{R}^p$ and $Y \subset \mathbb{R}^q$ be two multivariate Gaussian random variables with covariance matrix $C = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix}$. Then their mutual information is

$$
I(X,Y) = -\frac{1}{2} \log\left(\frac{\det C}{\det C_{11} \det C_{22}} \right) \tag{C.1.3}
$$

Proof. Let $Z = (X, Y)$. Without lost of generality lets assume that $\mathrm{E}[X] = \mathrm{E}[Y] = 0$.

$$
\begin{aligned}
I(X,Y) &= \int_{X,Y} \frac{1}{(2\pi)^{n/2}|C|^{1/2}} \exp\left(-\frac{1}{2}(x,y)^T C^{-1}(x,y)\right) \\
&\quad \times \log \frac{\frac{1}{(2\pi)^{n/2}|C|^{1/2}} \exp\left(-\frac{1}{2}(x,y)^T C^{-1}(x,y)\right)}{\frac{1}{(2\pi)^{n/2}|C_{11}|^{1/2}|C_{22}|^{1/2}} \exp\left(-\frac{1}{2}x^T C_{11}^{-1} x - \frac{1}{2}y^T C_{22}^{-1} y\right)} dxdy \\
&= \int_{X,Y} \frac{1}{(2\pi)^{n/2}|C|^{1/2}} \exp\left(-\frac{1}{2}(x,y)^T C^{-1}(x,y)\right) \\
&\quad \times -\frac{1}{2}\log\frac{|C|}{|C_{11}||C_{22}|}\left((x,y)^T C^{-1}(x,y) - x^T C_{11}^{-1} x - y^T C_{22}^{-1} y\right) dxdy \\
&= -\frac{1}{2}\log\frac{|C|}{|C_{11}||C_{22}|} - \frac{1}{2}\mathrm{E}_{X,Y}\left[(X,Y)^T C^{-1}(X,Y) - X^T C_{11}^{-1} X - Y^T C_{22}^{-1} Y\right] \\
&= -\frac{1}{2}\log\frac{|C|}{|C_{11}||C_{22}|} - \frac{1}{2}\mathrm{E}_{X,Y}\Big[\operatorname{Tr}\left((X,Y)^T C^{-1}(X,Y)\right) - \operatorname{Tr}\left(X^T C_{11}^{-1} X\right) \\
&\quad - \operatorname{Tr}\left(Y^T C_{22}^{-1} Y\right)\Big] \\
&= -\frac{1}{2}\log\frac{|C|}{|C_{11}||C_{22}|} - \frac{1}{2}\mathrm{E}_{X,Y}\Big[\operatorname{Tr}\left(C^{-1}(X,Y)(X,Y)^T\right) - \operatorname{Tr}\left(C_{11}^{-1} X X^T\right) \\
&\quad - \operatorname{Tr}\left(C_{22}^{-1} Y Y^T\right)\Big] \\
&= -\frac{1}{2}\log\frac{|C|}{|C_{11}||C_{22}|} - \frac{1}{2}\operatorname{Tr}\left(C^{-1}\mathrm{E}_{X,Y}\left[(X,Y)(X,Y)^T\right]\right) \\
&\quad + \frac{1}{2}\operatorname{Tr}\left(C_{11}^{-1}\mathrm{E}_{X,Y}\left[XX^T\right]\right) + \frac{1}{2}\operatorname{Tr}\left(C_{22}^{-1}\mathrm{E}_{X,Y}\left[YY^T\right]\right) \\
&= -\frac{1}{2}\log\frac{|C|}{|C_{11}||C_{22}|} - \frac{1}{2}\operatorname{Tr}\left(C^{-1}C\right) + \frac{1}{2}\operatorname{Tr}\left(C_{11}^{-1}C_{11}\right) + \frac{1}{2}\operatorname{Tr}\left(C_{22}^{-1}C_{22}\right) \\
&= -\frac{1}{2}\log\frac{|C|}{|C_{11}||C_{22}|} - \frac{1}{2}(p+q) + \frac{1}{2}p + \frac{1}{2}q \\
&= -\frac{1}{2}\log\frac{|C|}{|C_{11}||C_{22}|}
\end{aligned}
$$

□

C.1.3 HSIC

Proposition C.1.5. (HS norm of Tensor Products): The Hilbert-Schmidt inner product of two tensor products can be rewritten in the following way

$$\langle f \otimes g, h \otimes k \rangle_{HS} = \langle f, h \rangle_{\mathcal{F}} \langle g, k \rangle_{\mathcal{G}}$$

Proof. Let $u = \{u_1, u_2, ...\}$ and $v = \{v_1, v_2, ...\}$ be the orthonormal basis for $\mathcal{F}$ and $\mathcal{G}$ respectively. Let $f, h \in \mathcal{F}$ and $g, k \in \mathcal{G}$. Then from the definition of the HS norm, it follows that

$$\begin{aligned}
\langle f \otimes g, h \otimes k \rangle_{HS} &= \sum_{i,j} \langle (f \otimes g) v_i, u_j \rangle_{\mathcal{F}} \langle (h \otimes k) v_i, u_j \rangle_{\mathcal{F}} \\
&= \sum_{i,j} \langle f \langle g, v_i \rangle_{\mathcal{G}}, u_j \rangle_{\mathcal{F}} \langle h \langle k, v_i \rangle_{\mathcal{G}}, u_j \rangle_{\mathcal{F}} \\
&= \sum_{i} \left[\sum_{j} \langle f, u_j \rangle_{\mathcal{F}} \langle g, v_i \rangle_{\mathcal{G}} \langle h, u_j \rangle_{\mathcal{F}} \langle k, v_i \rangle_{\mathcal{G}} \right] \\
&= \sum_{i} \langle g, v_i \rangle_{\mathcal{G}} \langle k, v_i \rangle_{\mathcal{G}} \left[\sum_{j} \langle f, u_j \rangle_{\mathcal{F}} \langle h, u_j \rangle_{\mathcal{F}} \right] \\
&= \left(\sum_{i} \left\langle \sum_{n} \tilde{g}_n v_n, v_i \right\rangle_{\mathcal{G}} \left\langle \sum_{n} \tilde{k}_n v_n, v_i \right\rangle_{\mathcal{G}} \right) \\
&\quad \times \left(\sum_{j} \left\langle \sum_{n} \tilde{f}_n u_n, u_j \right\rangle_{\mathcal{F}} \left\langle \sum_{n} \tilde{h}_n u_n, u_j \right\rangle_{\mathcal{F}} \right) \\
&= \left(\sum_{i} \tilde{g}_i \tilde{k}_i \right) \left(\sum_{j} \tilde{f}_j \tilde{h}_j \right) \\
&= \langle f, h \rangle_{\mathcal{F}} \langle g, k \rangle_{\mathcal{G}}
\end{aligned}$$

The first equality follows from definition of Hilbert-Schmidt norm (3.4.2), the second equality follows from definition of tensor product (3.4.3). Here we wrote $f = \sum_n \tilde{f}_n u_n$ and similarly for g, h and k. □

Lemma C.1.6. (HSIC in terms of kernels): Hilbert-Schmidt Independence Criterion can be expressed in terms of kernels in the following way:

$$\begin{aligned}
\mathrm{HSIC}(p_{xy}, \mathcal{F}, \mathcal{G}) = \mathrm{E}_{x,y,x',y'} \Big[k(x, x') l(y, y') \Big] \\
- 2\mathrm{E}_{x,y} \Big[\mathrm{E}_{x'} \Big[k(x, x') \Big] \mathrm{E}_{y'} \Big[l(y, y') \Big] \Big] \\
+ \mathrm{E}_{x,x'} \Big[k(x, x') \Big] \mathrm{E}_{y,y'} \Big[l(y, y') \Big]
\end{aligned}$$

Proof.

$$\begin{aligned}
\|C_{xy}\|_{HS}^2 &= \langle \mathrm{E}_{xy}\Big[\phi_x \otimes \psi_y\Big] - \mu_x \otimes \mu_y, \mathrm{E}_{x'y'}\Big[\phi_{x'} \otimes \psi_{y'}\Big] - \mu_{x'} \otimes \mu_{y'} \rangle_{HS} \\
&= \mathrm{E}_{x,y,x',y'}\Big[\langle \phi_x \otimes \psi_y, \phi_{x'} \otimes \psi_{y'} \rangle_{HS}\Big] \\
&\quad - 2\mathrm{E}_{x,y}\Big[\langle \mu_x \otimes \mu_y, \phi_x \otimes \psi_y \rangle_{HS}\Big] \\
&\quad + \langle \mu_x \otimes \mu_y, \mu_x \otimes \mu_y \rangle_{HS} \\
&= \mathrm{E}_{x,y,x',y'}\Big[\langle \phi_x, \phi_{x'} \rangle_{\mathcal{F}} \langle \psi_y, \psi_{y'} \rangle_{\mathcal{G}}\Big] \\
&\quad - 2\mathrm{E}_{x,y}\Big[\langle \mu_x, \phi_x \rangle_{\mathcal{F}} \langle \mu_y, \psi_y \rangle_{\mathcal{G}}\Big] \\
&\quad + \langle \mu_x, \mu_x \rangle_{\mathcal{F}} \langle \mu_y, \mu_y \rangle_{\mathcal{G}} \\
&= \mathrm{E}_{x,y,x',y'}\Big[\langle \phi_x, \phi_{x'} \rangle_{\mathcal{F}} \langle \psi_y, \psi_{y'} \rangle_{\mathcal{G}}\Big] \\
&\quad - 2\mathrm{E}_{x,y}\Big[\mathrm{E}_{x'}\langle \phi_{x'}, \phi_x \rangle_{\mathcal{F}} \mathrm{E}_{y'}\langle \psi_{y'}, \psi_y \rangle_{\mathcal{G}}\Big] \\
&\quad + \mathrm{E}_x\mathrm{E}_{x'}\langle \phi_x, \phi_{x'} \rangle_{\mathcal{F}} \mathrm{E}_y\mathrm{E}_{y'}\langle \psi_y, \psi_{y'} \rangle_{\mathcal{G}} \\
&= \mathrm{E}_{x,y,x',y'}\Big[k(x,x')l(y,y')\Big] \\
&\quad - 2\mathrm{E}_{x,y}\Big[\mathrm{E}_{x'}\Big[k(x,x')\Big]\mathrm{E}_{y'}\Big[l(y,y')\Big]\Big] \\
&\quad + \mathrm{E}_{x,x'}\Big[k(x,x')\Big]\mathrm{E}_{y,y'}\Big[l(y,y')\Big]
\end{aligned}$$

The first equality follows from the linearity of the Hilbert-Schmidt norm. The second equality follows from Equation 3.4.4. The third equation follows from the definition of $\mu_x = \mathrm{E}_{x'}[\phi(x')]$ and $\mu_y = \mathrm{E}_{y'}[\phi(y')]$. The fourth equality follows from linearity of expectation and definition of $\langle \phi_x, \phi_{x'} \rangle_{\mathcal{F}} = k(x, x')$. □

Theorem C.1.7. Let E_{XY} denote the expectation taken over n independent samples (x_i, y_i) drawn from p_{XY}. Then

$$\mathrm{HSIC}(p_{XY}, \mathcal{F}, \mathcal{G}) = \mathrm{E}_{XY}\Big[\mathrm{HSIC}_b(X, Y, \mathcal{F}, \mathcal{G})\Big] + O(n^{-1}) \tag{C.1.4}$$

Proof. We begin by writing out the explicit form of $\operatorname{Tr} KHLH$

$$\begin{aligned}
\operatorname{Tr} KHLH &= \operatorname{Tr} K\Big(I - n^{-1}\mathbf{1}\mathbf{1}^T\Big)L\Big(I - n^{-1}\mathbf{1}\mathbf{1}^T\Big) \\
&= \operatorname{Tr}\Big(KL - n^{-1}K\mathbf{1}\mathbf{1}^TL - n^{-1}KL\mathbf{1}\mathbf{1}^T + n^{-2}K\mathbf{1}\mathbf{1}^TL\mathbf{1}\mathbf{1}^T\Big) \\
&= \operatorname{Tr} KL - n^{-1}\operatorname{Tr} K\mathbf{1}\mathbf{1}^TL - n^{-1}\operatorname{Tr} KL\mathbf{1}\mathbf{1}^T + n^{-2}\operatorname{Tr} K\mathbf{1}\mathbf{1}^TL\mathbf{1}\mathbf{1}^T \\
&= \operatorname{Tr} KL - 2n^{-1}\operatorname{Tr}\mathbf{1}^TLK\mathbf{1} + n^{-2}\operatorname{Tr}\mathbf{1}^TK\mathbf{1}\mathbf{1}^TL\mathbf{1} \\
&= \operatorname{Tr} KL - 2n^{-1}\mathbf{1}^TKL\mathbf{1} + n^{-2}\mathbf{1}^TK\mathbf{1}\mathbf{1}^TL\mathbf{1}
\end{aligned}$$

Now we will simplify each of the three terms separately. We need to take the expectation over n independent samples (x_i, y_i), we denote this by taking expectation over (X, Y) and (X', Y') where X and X' are independent and identically distributed, similarly for Y and Y'.

$$\begin{aligned}\mathrm{E}[\operatorname{Tr} KL] &= \mathrm{E}_{X,Y}\left[\sum_i K_{ii}L_{ii}\right] + \mathrm{E}_{X,Y,X',Y'}\left[\sum_{(i,j)\in i_2^n} K_{ij}L_{ji}\right] \\ &= O(n) + \frac{n!}{(n-2)!}\mathrm{E}_{X,Y,X',Y'}\Big[k(X,X')l(Y,Y')\Big]\end{aligned}$$

K_{ii} is a constant for all i (constant will depend on the choice of kernel: $K_{ii} = 1$ for Gaussian kernel, $K_{ii} = 0$ for Euclidean distance kernel).

$$\begin{aligned}\mathrm{E}[\mathbf{1}^T KL\mathbf{1}] = {} & \mathrm{E}_{X,Y}\left[\sum_i K_{ii}L_{ii}\right] + \mathrm{E}_{X,Y,X',Y'}\left[\sum_{(i,j)\in i_2^n}(K_{ii}L_{ij} + K_{ij}L_{jj})\right] \\ & + \mathrm{E}_{X,Y,X',Y''}\left[\sum_{(i,j,q)\in i_3^n} K_{ij}L_{jq}\right] \\ = {} & O(n) + O(n^2) + n(n-1)(n-2)\mathrm{E}_{X,Y,X',Y''}\Big[\mathrm{E}_{X'}\Big[k(X,X')\Big]\mathrm{E}_{Y''}\Big[l(Y,Y'')\Big]\Big] \\ = {} & O(n^2) + \frac{n!}{(n-3)!}\mathrm{E}_{X,Y,X',Y'}\Big[\mathrm{E}_{X'}\Big[k(X,X')\Big]\mathrm{E}_{Y'}\Big[l(Y,Y')\Big]\Big]\end{aligned}$$

There are n terms in the first summand, $n(n-1)$ terms in the second summand and $n(n-1)(n-2)$ terms in the third summand.

$$\begin{aligned}\mathrm{E}[\mathbf{1}^T K\mathbf{1}\mathbf{1}^T L\mathbf{1}] = {} & \mathrm{E}\Big[\sum_{i,j} K_{i,j}\sum_{i,j} L_{i,j}\Big] = \sum_{i,j,q,r}\mathrm{E}_{X,Y',X'',Y'''}\Big[K_{i,j}L_{q,r}\Big] \\ = {} & \mathrm{E}_{X,Y}\left[\sum_i K_{ii}L_{ii}\right] + \mathrm{E}_{X,Y,X',Y'}\left[\sum_{(i,j)\in i_2^n}(K_{ii}L_{ij} + K_{ij}L_{jj})\right] \\ & + \mathrm{E}_{X,Y,X',Y''}\left[\sum_{(i,j,q)\in i_2^n}(K_{ii}L_{jq} + K_{ij}L_{iq} + K_{ij}L_{qq})\right] \\ & + \sum_{(i,j,q,r)\in i_n^4}\mathrm{E}_{X,X',Y'',Y'''}\Big[K_{ij}L_{qr}\Big] \\ = {} & O(n^3) + \frac{n!}{(n-4)!}\mathrm{E}_{X,X'}\Big[k(X,X')\Big]\mathrm{E}_{Y'',Y'''}\Big[l(Y'',Y''')\Big]\end{aligned}$$

There are n, $n(n-1)$ and $n(n-1)(n-2)$ terms in the first, second and third summand respectively. The fourth summand has $n(n-1)(n-2)(n-3)$ terms.

Combining all three terms and using approximation $\frac{1}{n^k}\frac{n!}{(n-k-1)!} = 1 + O(n^{-1})$ we get:

$$\begin{aligned}\mathrm{E}\Big[\frac{1}{n^2}\operatorname{Tr} KHLH\Big] &= \mathrm{E}_{X,Y,X',Y'}\Big[k(X,X')l(Y,Y')\Big] + \mathrm{E}_{X,Y}\Big[\mathrm{E}_{X'}\Big[k(X,X')\Big]\mathrm{E}_{Y'}\Big[l(Y,Y')\Big]\Big] \\ &\quad + \mathrm{E}_{X,X'}\Big[k(X,X')\Big]\mathrm{E}_{X,X'}\Big[l(Y,Y')\Big] + O(n^{-1}) \\ &= \|C_{XY}\|^2_{HS} + O(n^{-1})\end{aligned}$$

□

Theorem C.1.8. (Mean of $\mathrm{HSIC}_b(X,Y)$): Under $\mathcal{H}_0$ the mean of $\mathrm{HSIC}_b(X,Y)$ is

$$\mathrm{E}(\mathrm{HSIC}_b(X,Y)) = n^{-1}\left(\mathrm{E}_{XY}kl - \mathrm{E}_X k\|\mu_Y\|^2 - \mathrm{E}_Y l\|\mu_X\|^2 + \|\mu_X\|^2\|\mu_Y\|^2\right)$$

Here $\mathrm{E}_X k = \mathrm{E}_X k(X,X)$ and $\mathrm{E}_Y l = \mathrm{E}_Y l(Y,Y)$.

Proof. First recall that the unbiased estimator of HSIC has form

$$\mathrm{HSIC}(X,Y) = \frac{1}{(n)_2}\sum_{(i,j)\in i_2^n} k_{ij}l_{ij} - 2\frac{1}{(n)_3}\sum_{(i,j,q)\in i_2^n} k_{ij}l_{iq} + \frac{1}{(n)_4}\sum_{(i,j,q,r)\in i_4^n} k_{ij}l_{qr}k_{ij}l_{iq}$$

where $(n)_k = \frac{n!}{(n-k)!}$ and $(i)_k^n$ are all k-tuples drawn from $1,2,...,n$ without replacement. The biased estimator of HSIC has form

$$\mathrm{HSIC}_b(x,y) = \frac{1}{n^2}\sum_{i,j=1}^n k_{ij}l_{ij} - 2\frac{1}{n^3}\sum_{i,j,q=1}^n k_{ij}l_{iq} + \frac{1}{n^4}\sum_{i,j,q,r=1}^n k_{ij}l_{qr}k_{ij}l_{iq}$$

Under null hypothesis (that is X and Y are independent) we have that $\mathrm{E}(\mathrm{HSIC}(X,Y)) = 0$. Therefore we have $\mathrm{E}(\mathrm{HSIC}_b(X,Y)) = \mathrm{E}(\mathrm{HSIC}_b(X,Y) - \mathrm{HSIC}(X,Y))$. Lets consider terms of this difference:

$$\begin{aligned}\frac{1}{n^2}\sum_{i,j=1}^n k_{ij}l_{ij} - \frac{1}{(n)_2}\sum_{(i,j)\in i_2^n} k_{ij}l_{ij} &= \frac{1}{n^2}\sum_{i=1}^n k_{ii}l_{ii} + \frac{n(n-1)-n^2}{n^2(n)_2}\sum_{(i,j)\in i_2^n} k_{ij}l_{ij} \\ &= \frac{1}{n^2}\sum_{i=1}^n k_{ii}l_{ii} - \frac{1}{n(n)_2}\sum_{(i,j)\in i_2^n} k_{ij}l_{ij}\end{aligned} \tag{C.1.5}$$

$$
\begin{aligned}
&\frac{1}{n^3}\sum_{i,j,q=1}^{n} k_{ij}l_{iq} - \frac{1}{(n)_3}\sum_{(i,j,q)\in i_3^n} k_{ij}l_{iq} \\
&\quad = \frac{1}{n^3}\sum_{i,j=1}^{n} k_{ii}l_{ii} + \frac{1}{n^3}\sum_{(i,j)\in i_2^n}(k_{ii}l_{ij} + k_{ij}l_{ii} + k_{ij}l_{ij}) \\
&\qquad + \frac{(n)_3 - n^3}{n^3(n)_3}\sum_{(i,j,q)\in i_3^m} k_{ij}l_{iq} \\
&\quad = \frac{1}{n^3}\sum_{i,j=1}^{n} k_{ii}l_{ii} + \frac{1}{n^3}\sum_{(i,j)\in i_2^n}(k_{ii}l_{ij} + k_{ij}l_{ii} + k_{ij}l_{ij}) \\
&\qquad - \frac{3}{n(n)_3}\sum_{(i,j,q)\in i_3^m} k_{ij}l_{iq} + O(n^{-5})
\end{aligned}
\tag{C.1.6}
$$

$$
\begin{aligned}
&\frac{1}{n^4}\sum_{i,j,q,r=1}^{n} k_{ij}l_{qr} - \frac{1}{(n)_4}\sum_{(i,j,q,r)\in i_4^n} k_{ij}l_{qr} \\
&\quad = \frac{1}{n^4}\sum_{i=1}^{n} k_{ii}l_{ii} + \frac{1}{n^4}\sum_{(i,j)\in i_2^n}(k_{ii}l_{ij} + k_{ij}l_{ii} + k_{ij}l_{ij}) \\
&\qquad + \frac{1}{n^4}\sum_{(i,j,q)\in i_3^n}(k_{ii}l_{jq} + 4k_{ij}l_{iq} + k_{ij}l_{qq}) + \frac{(n)_4 - n^4}{n^4(n)_4}\sum_{(i,j,q,r)\in i_4^n} k_{ij}l_{qr} \\
&\quad = \frac{1}{n^4}\sum_{i=1}^{n} k_{ii}l_{ii} + \frac{1}{n^4}\sum_{(i,j)\in i_2^n}(k_{ii}l_{ij} + k_{ij}l_{ii} + k_{ij}l_{ij}) \\
&\qquad + \frac{1}{n^4}\sum_{(i,j,q)\in i_3^n}(k_{ii}l_{jq} + 4k_{ij}l_{iq} + k_{ij}l_{qq}) - \frac{6}{n(n)_4}\sum_{(i,j,q,r)\in i_4^m} k_{ij}l_{qr} + O(n^{-6})
\end{aligned}
\tag{C.1.7}
$$

There is 4 before term $k_{ij}l_{iq}$ in Equation C.1.7 as there are 4 combinations of indices i, j, q, r in $k_{ij}l_{qr}$ that two indices would agree and the other two would differ: $i = q$, $i = r$, $j = q$ and $j = r$. Combining terms from Equations C.1.5-C.1.7 we get

$$
\begin{aligned}
\mathrm{HSIC}_b(x,y) - \mathrm{HSIC}(x,y) &= \left(\frac{1}{n^2} - \frac{2}{n^3} + \frac{1}{n^4}\right)\sum_{i=1}^{n} k_{ii}l_{ii} \\
&+ \left(-2\frac{1}{n^3} + \frac{1}{n^4}\right)\sum_{(i,j)\in i_2^n}(k_{ii}l_{ij} + k_{ij}l_{ii}) \\
&+ \frac{1}{n^4}\sum_{(i,j,q)\in i_3^n}(k_{ii}l_{jq} + k_{ij}l_{qq}) + \left(-\frac{1}{n(n)_2} - \frac{2}{n^3} + \frac{1}{n^4}\right)\sum_{(i,j)\in i_2^n} k_{ij}l_{ij} \\
&+ \left(4\frac{1}{n^4} + 2\frac{3}{n(n)_3}\right)\sum_{(i,j,q)\in i_3^n} k_{ij}l_{iq} - \frac{6}{n(n)_4}\sum_{(i,j,q,r)\in i_4^n} k_{ij}l_{qr} \qquad \text{(C.1.8)} \\
&\approx \frac{1}{n^2}\sum_{i=1}^{n} k_{ii}l_{ii} - \frac{2}{n(n)_2}\sum_{(i,j)\in i_2^n}(k_{ii}l_{ij} + k_{ij}l_{ii}) \\
&+ \frac{1}{n(n)_3}\sum_{(i,j,q)\in i_3^n}(k_{ii}l_{jq} + k_{ij}l_{qq}) - \frac{3}{n(n)_2}\sum_{(i,j)\in i_2^n} k_{ij}l_{ij} \\
&+ \frac{10}{n(n)_3}\sum_{(i,j,q)\in i_3^n} k_{ij}l_{iq} - \frac{6}{n(n)_4}\sum_{(i,j,q,r)\in i_4^n} k_{ij}l_{qr}
\end{aligned}
$$

Approximation is taken by dropping the lower order terms. Now we take the expectation of Equation C.1.8. We use notation $\mathrm{E}_X[k(X,X)] = \mathrm{E}_X k$, $\mathrm{E}_{XYY'}[k(X,X)l(Y,Y')] = \mathrm{E}_{XYY'}kl$, where X and X' are independent and identically distributed.

$$
\begin{aligned}
n\mathrm{E}(\mathrm{HSIC}_b(X,Y)) &= n\mathrm{E}(\mathrm{HSIC}_b(X,Y) - \mathrm{HSIC}(X,Y)) \\
&= \mathrm{E}_{XY}kl - 2\left(\mathrm{E}_{XYY'}kl + \mathrm{E}_{XX'Y}kl\right) \\
&\quad + \mathrm{E}_{XY'Y''}kl + \mathrm{E}_{XX'Y''}kl \\
&\quad - 3\mathrm{E}_{XX'YY'}kl + 10\mathrm{E}_{XX'YY''}kl - 6\mathrm{E}_{XX'}k\mathrm{E}_{YY'}l
\end{aligned}
$$

At independence $\mathrm{HSIC}(X,Y) = 0$ and three last terms merge; as X, X', Y and Y' are all independent we can factorize expectation as $\mathrm{E}_{XX'YY''}kl = \mathrm{E}_{XX'}k\mathrm{E}_{YY''}l$. We get

$$
\mathrm{E}(\mathrm{HSIC}_b(X,Y)) = n^{-1}\left(\mathrm{E}_{XY}kl - \mathrm{E}_X k\|\mu_Y\|^2 - \mathrm{E}_Y l\|\mu_X\|^2 + \|\mu_X\|^2\|\mu_Y\|^2\right)
$$

where $\|\mu_X\|^2 = \mathrm{E}_{XX'}[k(X,X')]$ and $\|\mu_Y\|^2 = \mathrm{E}_{YY'}[l(Y,Y')]$.

□

Theorem C.1.9. (Variance of $\mathrm{HSIC}_b(X,Y)$): Under $\mathcal{H}_0$ the variance of $\mathrm{HSIC}_b(X,Y)$ is

$$
\mathrm{Var}(\mathrm{HSIC}_b(X,Y)) = \frac{2(n-4)(n-5)}{(n)_4}\|C_{XX}\|_{HS}^2\|C_{YY}\|_{HS}^2
$$

Here $\|C_{XX}\|^2_{HS} = \frac{1}{n^2}\operatorname{Tr} KHKH + O(n^{-1})$.

Proof. We first express $\mathrm{HSIC}_b(x, y)$ as

$$\mathrm{HSIC}_b(x, y) = \frac{1}{m^4} \sum_{(i,j,q,r)\in i_4^m} h_{ijqr} \tag{C.1.9}$$

where

$$h_{ijqr} = \frac{1}{4!} \sum_{(t,u,v,w)\in\mathcal{P}(i,j,q,r)} k_{tu}l_{tu} - 2k_{tu}l_{tv} + k_{tu}l_{vw}$$

here $\mathcal{P}(i, j, q, r)$ denotes all the permutations of (i, j, q, r).
We start by computing the variance of the symmetric U-statistics $\mathrm{HSIC}_s(x, y)$ associated with C.1.9

$$\mathrm{HSIC}_s(x, y) = \frac{1}{(m)_4} \sum_{(i,j,q,r)\in i_4^m} h_{ijqr}$$

Using the symmetry of k_{ij} and l_{ij} in their index orderings we can make the following simplification

$$\begin{aligned} h_{ijqr} = \frac{1}{6}[&k_{ij}(l_{ij} + l_{qr}) + k_{iq}(l_{iq} + l_{jr}) + k_{ir}(l_{ir} + l_{jq}) \\ &+ k_{jq}(l_{jq} + l_{ir}) + k_{jr}(l_{jr} + l_{iq}) + k_{qr}(l_{qr} + l_{ij})] \\ &- \frac{1}{12} \sum_{(t,u,v)\in\mathcal{P}(i,j,q,r)} k_{tu}[l_{tv} + l_{uv}] \end{aligned}$$

Summation in the last term is taken over all ordered triples (t, u, v) selected without replacement from (i, j, q, r). Recall that $\mathrm{HSIC}_s(x, y)$ is a U-statistic. It follows from Theorem C.2.1 that variance of $\mathrm{HSIC}_s(X, Y)$ is

$$\operatorname{Var}(\mathrm{HSIC}_s(X, Y)) = \binom{n}{4}^{-1} \sum_{k=1}^{4} \binom{4}{k}\binom{n-4}{4-k}\zeta_k$$

we only need to consider

$$\zeta_1 = \mathrm{E}_{i,j,q}\Big[(\mathrm{E}_r[h_{ijqr}])^2\Big] \text{ and } \zeta_2 = \mathrm{E}_{i,j}\Big[(\mathrm{E}_{q,r}[h_{ijqr}])^2\Big]$$

as other terms decay faster than ζ_2 and can be neglected. Under the null hypothesis we have degeneracy $\zeta_1 = 0$. So (C.1.3) simplifies to

$$\operatorname{Var}(\mathrm{HSIC}_s(X, Y)) = \frac{72(n-4)(n-5)}{n(n-1)(n-2)(n-3)}\zeta_2 + O(n^{-3})$$

To make expressions easier to follow we introduce new notation: $\mathrm{E}_{q,r} := \mathrm{E}_{X_q Y_q X_r Y_r}$. We continue by finding $\mathrm{E}_{q,r}[h_{ijqr}]$. Using the fact that X and Y are independent under the null hypothesis, we get

$$\mathrm{E}_{q,r}\Big[k_{tu}l_{vw}\Big] = \mathrm{E}_{q,r}\Big[k_{tu}\Big]\mathrm{E}_{q,r}\Big[l_{vw}\Big], \text{ for all } (t,u,v,w) \in \mathcal{P}(i,j,q,r)$$

Also for all distinct i, j, q and r

$$\mathrm{E}_{q,r}\Big[k_{tu}\Big] = \begin{cases} k_{ij}, & \text{if } \{t,u\} = \{i,j\} \\ \mathrm{E}_X k(x_i, X), & \text{if } t = i \text{ and } u \in \{q,r\} \\ \mathrm{E}_{XX'} k(X,X'), & \text{if } \{t,u\} = \{q,r\} \end{cases}$$

To make expressions easier to follow we introduce new notation: $\mathrm{E}_X k_i := \mathrm{E}_X[k(x_i, X)]$ and $\mathrm{E}_{XX'}k := \mathrm{E}_{XX'}[k(X,X')]$.

$$\begin{aligned} 12\mathrm{E}_{q,r}[h_{ijqr}] &= k_{ij}(2l_{ij} + 2\mathrm{E}_{YY'}l - 2\mathrm{E}_Y l_i - 2\mathrm{E}_Y l_j) \\ &\quad - 2\mathrm{E}_X k_i(l_{ij} + \mathrm{E}_{YY'}l - \mathrm{E}_Y l_i - \mathrm{E}_Y l_j) \\ &\quad - 2\mathrm{E}_X k_j(l_{ij} + \mathrm{E}_{YY'}l - \mathrm{E}_Y l_i - \mathrm{E}_Y l_j) \\ &\quad + \mathrm{E}_{XX'}k(2l_{ij} + 2\mathrm{E}_{YY'}l - 2\mathrm{E}_Y l_i - 2\mathrm{E}_Y l_j) \\ &= 2(k_{ij} - \mathrm{E}_X k_i - \mathrm{E}_X k_j + \mathrm{E}_{XX'}k)(l_{ij} - \mathrm{E}_Y l_i - \mathrm{E}_Y l_j + \mathrm{E}_{YY'}l) \end{aligned}$$

Again using the assumption that X and Y are independent, the expectation $\mathrm{E}_{i,j}\left[\mathrm{E}_{q,r}[h_{ijqr}]\right]^2$ factorises into a product of expectation over X and over Y. The first is

$$\begin{aligned} &\mathrm{E}_{i,j}\Big[k_{ij} - \mathrm{E}_X k_i - \mathrm{E}_X k_j + \mathrm{E}_{XX'}k\Big]^2 \\ &\quad = \mathrm{E}_{i,j}\Big[\langle \phi_{X_i}, \phi_{X_j}\rangle - \langle \phi_{X_i}, \mu_X\rangle - \langle \mu_X, \phi_{X_j}\rangle + \langle \mu_X, \mu_X\rangle\Big]^2 \\ &\quad = \mathrm{E}_{i,j}\langle \phi_{X_i} - \mu_X, \phi_{X_j} - \mu_X\rangle_{\mathcal{F}}^2 \\ &\quad = \mathrm{E}_{ij}\langle (\phi_{X_i} - \mu_X)\otimes(\phi_{X_i} - \mu_X), (\phi_{X_j} - \mu_X)\otimes(\phi_{X_j} - \mu_X)\rangle_{HS} \\ &\quad = \|C_{XX}\|_{HS}^2 \end{aligned}$$

The second is $\|C_{YY}\|_{HS}^2$ by symmetry. Combining these expressions, we get

$$\mathrm{Var}(\mathrm{HSIC}_s(X,Y)) = \frac{2(n-4)(n-5)}{(n)_4}\|C_{XX}\|_{HS}^2\|C_{YY}\|_{HS}^2 + O(n^{-3}) \qquad \text{(C.1.10)}$$

The variance of $\mathrm{HSIC}_b(X,Y)$ is identical, since the additional terms that arise from the bias vanish faster than the leading terms retained in (C.1.10).

□

C.2 U-statistics

Results about the asymptotic behaviour of the Hilbert-Schmidt Independence Criterion relies heavily on the concept of U-statistics. In this section we present some of the essential definitions and properties of the U-statistics. We are following the exposition from the lecture notes by Ferguson (2005), for a more detailed presentation refer to Lee (1990) and Denker (1985).

Definition C.2.1. (Estimable parameter): Let $\mathcal{P}$ be a family of probability measures on an arbitrary measurable space. Let $\theta(P)$ denote a real-valued function defined for $P \in \mathcal{P}$. We say that $\theta(P)$ is an estimable parameter within $\mathcal{P}$, if for some integer m there exists an unbiased estimator of $\theta(P)$ based on m i.i.d. random variables distributed according to P; that is, if there exists a real-valued measurable function $h(x_1, ..., x_m)$ such that

$$\mathrm{E}_P(h(X_1, ..., X_m)) = \theta(P) \text{ for all } P \in \mathcal{P},$$

when $X_1, ..., X_m$ are i.i.d. with distribution P. The smallest integer m with this property is called the degree of $\theta(P)$.

Definition C.2.2. (U-statistic): For a real-valued measurable function, $h(x_1, ..., x_m)$ and for a sample, $X_1, ..., X_n$, of size $n \geq m$ from a distribution p_X, a U-statistic with kernel h of order k is defined as

$$U_n = U_n(h) = \binom{n}{m}^{-1} \sum_{C_{m,n}} h\left(X_{i_1}, ..., X_{i_m}\right) \tag{C.2.1}$$

where the summation is over the set $C_{m,n}$ of all $\binom{n}{m}$ combinations of m integers, $i_1 < ... < i_m$ chosen from $\{1, 2, ..., n\}$ without replacement.

We define

$$\begin{aligned} h_k(x_1, ..., x_k) &= \mathrm{E}[h(X_1, ..., X_m) \mid X_1 = x_1, ..., X_k = x_k] \\ &= \mathrm{E}[h(x_1, ..., x_k, X_{k+1}, ..., X_m)] \end{aligned}$$

Theorem C.2.1 (Ferguson (2005))**.** (Variance of U-statistics): For a U-statistic U_n given by Equation C.2.1 with $\mathrm{E}[h(X_1, ..., X_m)]^2 < \infty$,

$$\mathrm{Var}(U_n) = \binom{n}{m}^{-1} \sum_{k=1}^{m} \binom{m}{k} \binom{n-m}{m-k} \zeta_k$$

where

$$\zeta_k = \mathrm{Var}(h_k(X_1, ..., X_k))$$

If $\zeta_m^2 < \infty$, then $\mathrm{Var}(U_n) \sim m^2\zeta_1/n$ for large n.

Proof.

$$\begin{aligned}\mathrm{Var}(U_n) &= \mathrm{Var}\left(\binom{n}{m}^{-1} \sum_{C_{m,n}} h(X_{i_1}, ..., X_{i_m})h(X_{i_1}, ..., X_{i_m})\right) \\ &= \binom{n}{m}^{-2} \sum_{i \in C_{m,n}} \sum_{j \in C_{m,n}} \mathrm{Cov}\Big(h(X_{i_1}, ..., X_{i_m}), h(X_{j_1}, ..., X_{j_m})\Big)\end{aligned}$$

Proposition C.2.2 (Ferguson (2005))**.** For $(i_1, ..., i_m)$ and $(j_1, ..., j_m)$ in $C_{m,n}$ and $X_{i_1}, ..., X_{i_m}, X_{j_1}, ..., X_{j_m}$ i.i.d.,

$$\begin{aligned}&\mathrm{Cov}(h(X_{i_1}, ..., X_{i_m}), h(X_{j_1}, ..., X_{j_m})) \\ &\quad = \mathrm{Cov}(h_k(X_1, ..., X_k), h(X_1, ..., X_m)) = \zeta_k\end{aligned}$$

where k is the number of common elements between $(i_1, ..., i_m)$ and $(j_1, ..., j_m)$.

Proof.

$$\begin{aligned}&\mathrm{Cov}(h(X_{i_1}, ..., X_{i_m}), h(X_{j_1}, ..., X_{j_m})) \\ &\quad = \mathrm{Cov}(h(X_1, ..., X_k, X_{k+1}, ..., X_m), h(X_1, ..., X_k, X'_{k+1}, ..., X'_m)) \\ &\quad = \mathrm{E}\Big[(h(X_1, ..., X_k, X_{k+1}, ..., X_m))(h(X_1, ..., X_k, X'_{k+1}, ..., X'_m))\Big] \\ &\qquad - \mathrm{E}\Big[(h(X_1, ..., X_k, X_{k+1}, ..., X_m))\Big]\mathrm{E}\Big[(h(X_1, ..., X_k, X'_{k+1}, ..., X'_m))\Big] \\ &\quad = \mathrm{E}\Big[h(X_1, ..., X_k, X_{k+1}, ..., X_m)h(X_1, ..., X_k, X'_{k+1}, ..., X'_m)\Big] - \theta^2 \\ &\quad = \mathrm{E}_{X_1,...,X_k}\Big[\mathrm{E}\Big[h(X_1, ..., X_k, X_{k+1}, ..., X_m)h(X_1, ..., X_k, X'_{k+1}, ..., X'_m)|X_1, ..., X_k\Big]\Big] - \theta^2 \\ &\quad = \mathrm{E}\Big[h_k(X_1, ..., X_k)h_k(X_1, ..., X_k)\Big] - \theta^2 \\ &\quad = \zeta_k\end{aligned}$$

Here $\theta = \mathrm{E}h(X_1, ..., X_m)$. We get the first equality using the fact that $(i_1, ..., i_m)$ and $(j_1, ..., j_m)$ have k elements in common and h is symmetric. Here $X_{k+1}, ..., X_m, X'_{k+1}, ..., X'_m$

are i.i.d. We get the third equality by conditioning on $X_1, ..., X_k$ and using the definition of h_k, as then two terms become independent.
The same argument shows that $\text{Cov}(h_k(X_1, ..., X_k), h(X_1, ..., X_m)) = \zeta_k$.

□

The number of m-tuples $(i_1, ..., i_m)$ and $(j_1, ..., j_m)$ having exactly k elements in common is $\binom{n}{m}\binom{m}{k}\binom{n-m}{m-k}$ as there are $\binom{n}{m}$ ways to choose $(i_1, ..., i_m)$, there are $\binom{m}{k}$ ways to choose k elements in this m-tuple that will also be in $(j_1, ..., j_m)$ and finally $\binom{n-m}{m-k}$ ways to choose other $m-k$ elements for $(j_1, ..., j_m)$ out of remaining $n-m$ elements. Therefore,

$$\begin{aligned}\text{Var}(U_n) &= \binom{n}{m}^{-2} \sum_{k=0}^{m} \binom{n}{m}\binom{m}{k}\binom{n-m}{m-k}\zeta_k \\ &= \binom{n}{m}^{-1} \sum_{k=0}^{m} \binom{m}{k}\binom{n-m}{m-k}\zeta_k\end{aligned}$$

If $\zeta_m < \infty$, then $\zeta_i < \infty$ for all $i < m$. For large n, the first term of the sum dominates since it is of the largest order. The coefficient of ζ_1 is $m\binom{n-m}{m-1}/\binom{n}{m} \sim m^2/n$.

□

Theorem C.2.3 (Ferguson (2005))**.** If $\zeta_m < \infty$, then

$$\sqrt{n}(U_n - \theta) \to \mathcal{N}(0, m^2\zeta_1)$$

Proof. Let

$$U_n^* = \frac{m}{n}\sum_{k=1}^{n}(h_1(X_k) - \theta)$$

Then since $m(h_1(X_k) - \theta)$ are i.i.d. with mean 0 and variance $m^2\zeta_1$, the central limit theorem implies that $\sqrt{n}U_n^* \to \mathcal{N}(0, m^2\zeta_1)$. We continue by showing that $\sqrt{n}U_n^*$ is asymptotically equivalent to $\sqrt{n}(U_n - \theta)$ and so they both have same limiting distribution. It is enough to show that $n\text{E}(U_n^* - (U_n - \theta))^2 \to 0$.

$$n\text{E}(U_n^* - (U_n - \theta))^2 = n\text{Var}(U_n^*) - 2n\text{Cov}(U_n^*, U_n) + n\text{Var}(U_n)$$

$$
\begin{aligned}
n\text{Var}(U_n^*) &= m^2\zeta_1 \\
n\text{Var}(U_n) &\to m^2\zeta_1 \\
n\text{Cov}(U_n^*, U_n) &= \frac{m}{\binom{n}{m}} \sum_{k=1}^{n} \sum_{j \in C_{m,n}} \text{Cov}(h_1(X_k), h(X_{j_1}, ..., X_{j_m})) \\
&= \frac{m}{\binom{n}{m}} n \binom{n-1}{m-1} \zeta_1 = m^2\zeta_1
\end{aligned}
$$

As by Theorem C.2.1

$$
\text{Cov}(h_1(X_k), h(X_{j_1}, ..., X_{j_m})) = \begin{cases} 0, & \text{if } k \notin \{j_1, ..., j_m\} \\ \zeta_1, & \text{if } k \in \{j_1, ..., j_m\} \end{cases}
$$

There are n choices for k and for any fixed k there are $\binom{n-1}{m-1}$ choices of set $\{j_1, .., j_m\}$ containing k.

□

Definition C.2.3. We say that a U-statistic has a degeneracy of order k if $\zeta_1 = ... = \zeta_k = 0$ and $\zeta_{k+1} > 0$.

Theorem C.2.4 (Ferguson (2005))**.** Let U_n be the U-statistic associated with a symmetric kernel of degree 2, degeneracy of order 1, and expectation θ. Then

$$
n(U_n - \theta) \to \sum_{k=1}^{\infty} \lambda_k (Z_k^2 - 1)
$$

where $Z_1, Z_2, ...$ are independent $\mathcal{N}(0, 1)$ and $\lambda_1, \lambda_2, ...$ are the eigenvalues of

$$
h(x_1, x_2) - \theta = \sum_{k=1}^{\infty} \lambda_k \psi_k(x_1) \psi_k(x_2)
$$

where ϕ_k's are the orthonormal sequences, i.e.

$$
\text{E}\psi_i(X_1)\psi_j(X_2) = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{if } i \neq j. \end{cases}
$$

C.3 Approximating Mutual information using Gaussian random variables

Let X^G and Y^G denote Gaussian random variables that have the same covariance structure as X and Y. Namely:

$$E(XY^T) = P, E(X) = p_x, E(Y) = p_y, E(XX^T) = D_{p_x}, E(YY^T) = D_{p_y}$$

Here $D_{p_x} = diag(p_x)$ and $D_{p_y} = diag(p_y)$.

Then $I^G = I(X^G, Y^G) = -\frac{1}{2}\log\det\left(I - CC^T\right) = -\frac{1}{2}\log\det\left(I - C^TC\right)$. Where $C = D_{p_x}^{-1/2}\left(P - p_x p_y^T\right)D_{p_y}^{-1/2}$ (the correlation matrix).Then 'close to independence', we can assume $P_{ij} = p_{xi}p_{yj}\left(1 + \epsilon_{ij}\right)$ (here ϵ has small norm). Then we can rewrite C as $C = D_{p_x}^{-1/2}\left(P - p_x p_y^t\right)D_{p_y}^{-1/2} = D_{p_x}^{1/2}\epsilon D_{p_y}^{1/2}$ and so C also has small norm. So we have:

$$I^G = -\frac{1}{2}\log\det\left(I - CC^T\right) \approx \frac{1}{2}\operatorname{Tr}\left(CC^T\right)$$

And so we obtain:

$$I^G \approx \frac{1}{2}\operatorname{Tr}\left(D_{p_x}\epsilon D_{p_y}\epsilon^T\right) = \frac{1}{2}\sum_{ij}\epsilon_{ij}^2 p_{xi}p_{yj}$$

For non-Gaussian random variables X and Y we expand the mutual information $I = I(X, Y)$

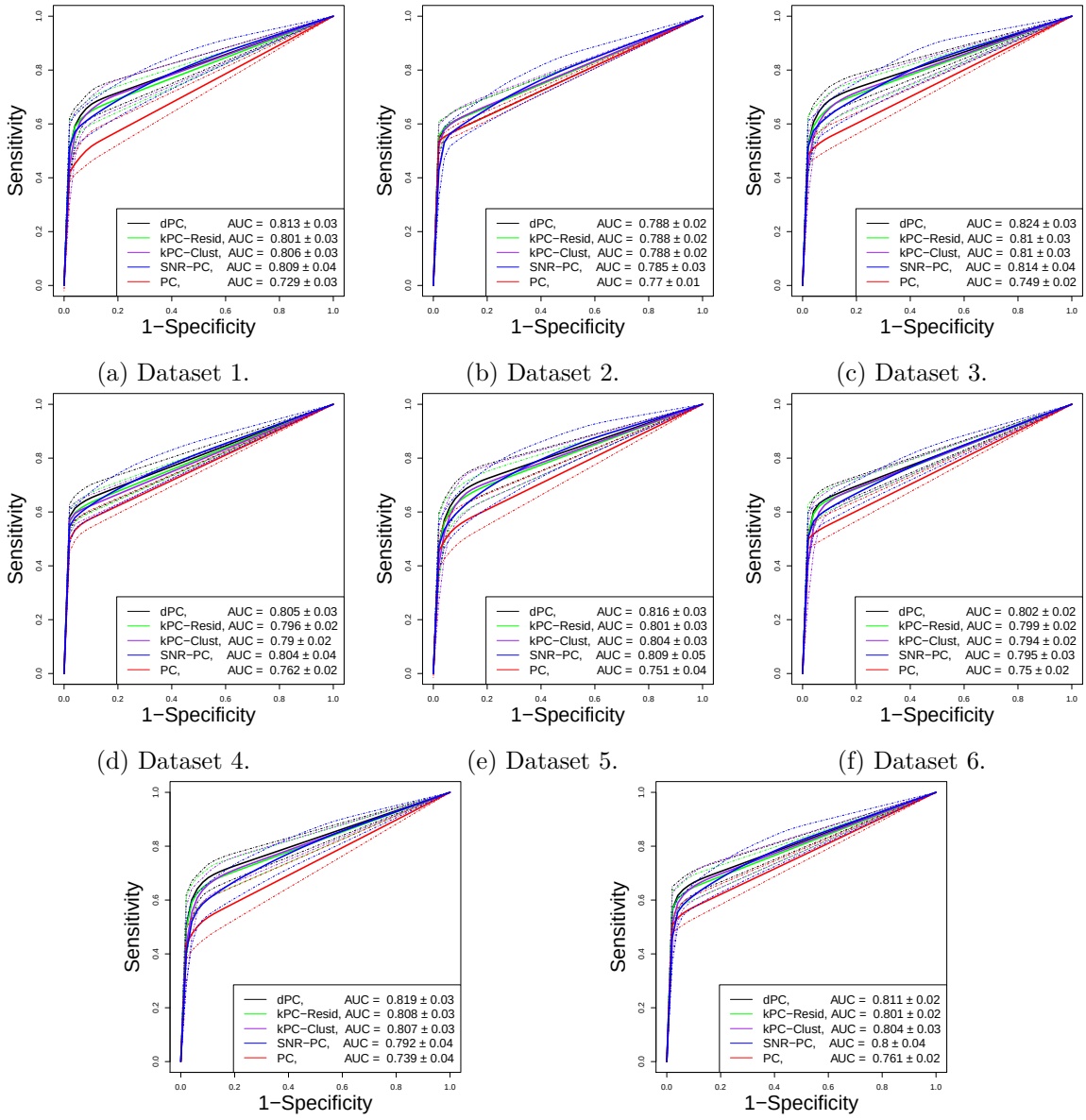
$$I = \sum_{ij} p_{xi}p_{yj}\left(1 + \epsilon_{ij}\right)\log\frac{p_{xi}p_{yj}\left(1 + \epsilon_{ij}\right)}{p_{xi}p_{yj}} \overset{1}{\approx} \sum_{ij} p_{xi}p_{yj}\left(\epsilon_{ij} + \frac{1}{2}\epsilon_{ij}^2\right) \overset{2}{=} \frac{1}{2}\sum_{ij}\epsilon_{ij}^2 p_{xi}p_{yj}$$

1. Using Taylor expansion: $\left(1 + \epsilon_{ij}\right)\log\left(1 + \epsilon_{ij}\right) \approx \epsilon_{ij} + \frac{1}{2}\epsilon_{ij}^2$
2. $\sum_{ij} p_{xi}p_{yj}\epsilon_{ij} = \sum_{ij} p_{xi}p_{yj}\left(1 + \epsilon_{ij}\right) - \sum_{ij} p_{xi}p_{yj} = \sum_{ij} P_{ij} - \sum_{ij} p_{xi}p_{yj} = 1 - 1 = 0$

So we conclude that for any random variables X and Y their mutual information I is the same up to the second order to the mutual information I^G of Gaussian random variables X^G and Y^G with same covariance structure as X and Y.

Appendix D

Kernel PC

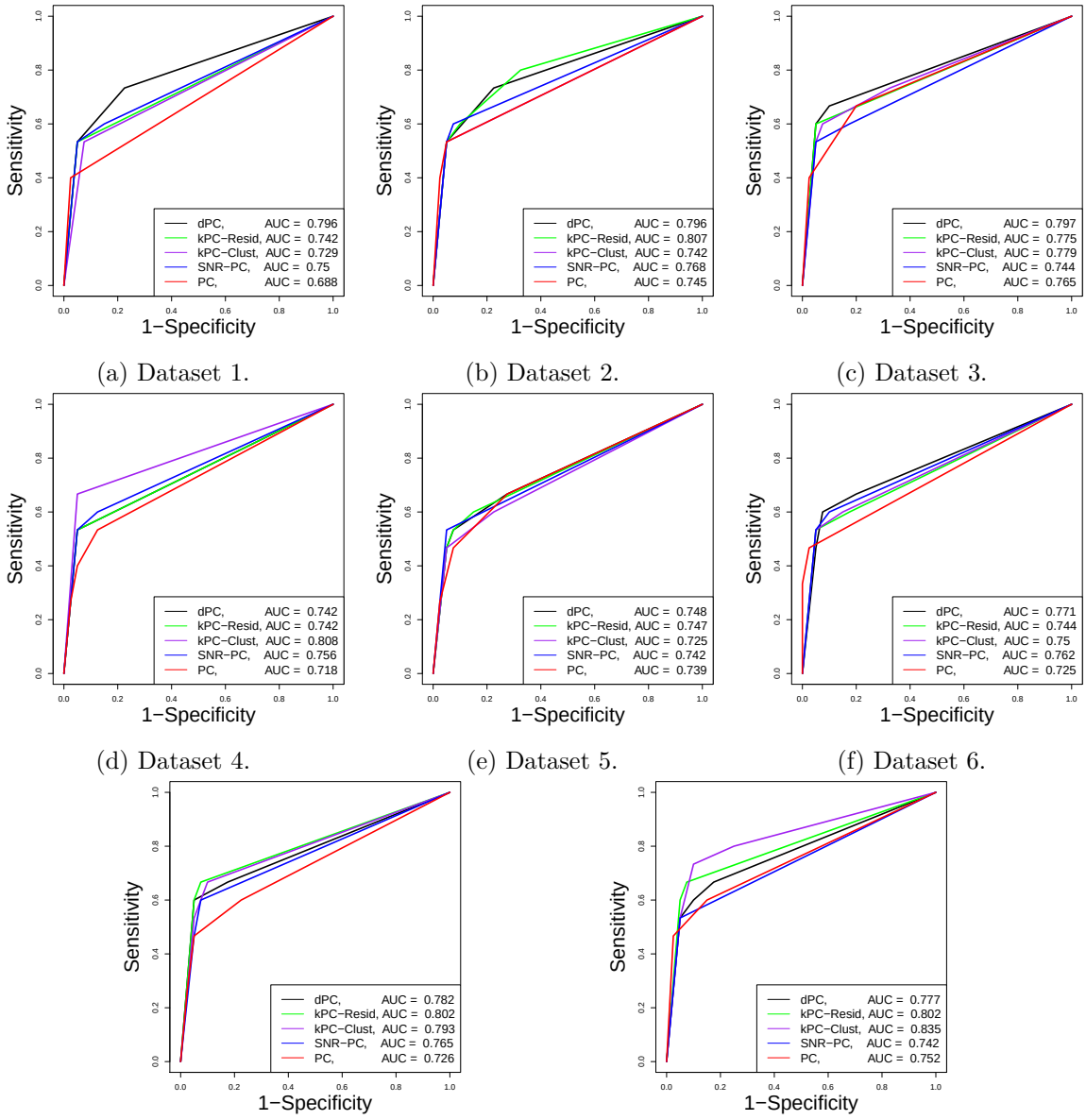


(a) Dataset 1. (b) Dataset 2. (c) Dataset 3.

(d) Dataset 4. (e) Dataset 5. (f) Dataset 6.

(g) Dataset 7. (h) Dataset 8.

Fig. D.1 ROC curves to compare kPCs, dPC, SNR-PC and PC algorithms on the data simulated from single-cell data from Sachs et al. (2005).



(a) Dataset 1. (b) Dataset 2. (c) Dataset 3.

(d) Dataset 4. (e) Dataset 5. (f) Dataset 6.

(g) Dataset 7. (h) Dataset 8.

Fig. D.2 ROC curves to compare kPCs, dPC, SNR-PC and PC algorithms on all 8 datasets from Sachs et al. (2005).