Causal Inference and Conditional Independence Testing with RCoT

Mayank Agarwal [1], Abhay H. Kashyap [1,*], G. Shobha [1], Jyothi Shetty [1], and Roger Dev [2]

[1] Department of Computer Science, R. V. College of Engineering, Bangalore, India
[2] Lexis Nexis Risk Solution, Alpharetta, USA; Email: Roger.Dev@lexisnexisrisk.com (R.D.)
*Correspondence: abhayhkashyap01@gmail.com (A.H.K.)

Abstract—**Conditional Independence (CI) testing is a crucial operation in causal model discovery and validation. Effectively performing this requires a linearly scalable and robust algorithm and its implementation. Previous techniques, such as cross-correlation, a linear method; Kernel Conditional Independence Test (KCIT,) and a kernel-based algorithm, do not scale well with dataset size and pose a bottleneck for CI algorithms. An improved version of kernel-based algorithms which use linear mapping to decrease computational time is the Randomized conditional Correlation Test (RCoT) and Randomized Conditional Independence Test (RCIT). This paper describes their use and implementation in Python. This paper then compares the time complexity of the RCoT algorithm with a previously implemented Discretization-based algorithm Probspace. The results show that the accuracy of the previous and current models is similar, but the time taken to get these results has been reduced by 50%. The implemented algorithm takes about 3s to run the testcases (the data used and testcases generated are described in Section IV-C).**

Keywords—**causal inference, conditional independence testing, Randomized conditional Correlation Test (RCoT) algorithm, Lindsay-Pilla-Basaky approximation, Fourier features**

I. Introduction

Causality is the study of the causes and effects of events in the environment. It is fundamental in gaining information about the environment to model it and predict further events. Causal models are used to satisfactorily model causal relationships between variables (cause/effect). A Causal Model is a directed graph of the causal relationships of random variables and has the following components [1]:

- A set of nodes (*W, X, Y, Z*) representing random variables.
- A set of directed edges ($W \rightarrow Y$, $W \rightarrow X$, $Y \rightarrow Z$, $X \rightarrow Z$) between pairs of nodes, each edge regarded as the hypothesis that the node to which the edge is incident depends on the other node if values of all other random variables were fixed.
- Joint probability distribution over the possible values of all the variables.

Manuscript received August 29, 2022; revised September 29, 2022; accepted October 20, 2023; published June 1, 2023.

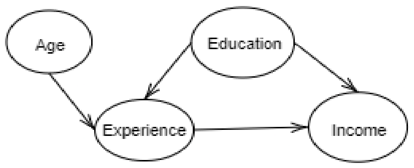


Figure 1. Example of causal model showing factors effecting the income of a person.

Fig. 1 shows the observed factors that directly or indirectly affect a person's income. This causal model shows most of the causality concepts like common affect, education and age affect the experience in the industry, common cause, experience, and income is directly affected by education. The purpose of creating or "fitting" a causal model is to derive reasonable inferences from the relations in the model. This process is termed causal inference. Causal inference is the process of separating causal from non-causal influences between variables in a particular phenomenon. Causal science makes heavy use of Conditional Independence testing to discover causal relationships and validate causal assumptions. Conditional Independence Tests are methods of finding the dependency of one random variable with respect to another when one or more random variables are conditioned on. It involves measuring the linear and non-linear correlation between the variables. Some popular methods used for Independence testing are Regression and permutation-based tests [2]. The cross-correlation and partial cross-correlation methods are widely used linear methods as a test for conditional independence [3]. Most of these algorithms make assumptions about the data or the conditioning variables that are hard to justify in the real world and may lead to incorrect approximations, such as linearity or Gaussianity [4].

Recently, algorithms used for independence testing of random variables are kernel-based methods, that detect non-linear dependencies. They counter the computational bottleneck, which is a disadvantage of using non-kernel-based independence testing algorithms [2]. Kernel-based algorithms perform well and give accurate results for both conditional and non-conditional independence testing. But in the case of CI testing, the dimension of the conditioning variable dictates the time complexity, and thus, the algorithms' time complexity scales cubically with the conditioning set [5]. This paper discusses a kernel-based algorithm, RCoT and its first implementation in Python. The results of the experiment are then compared with previously used non-kernel-based algorithms for Independence Testing.

II. LITERATURE REVIEW

Causal interventions or randomized experiments are used to identify causal relationships among a set of random variables and generate a causal model for them. This method, in many cases, is expensive, time-consuming, or even impossible [6]. Alternatively, there are two ways for generating the causal model [7]:

1. One way is to assume relationships between the random variables and, by verifying with the data, discover the causal model that generated it, known as Causal Discovery. These methods try revealing causal information by analyzing purely observational data [8].
2. The second method uses prior knowledge to create a causal model, but it may be missing crucial information, therefore, the model is validated with data to test whether the knowledge of the domain fits the data that is observed. Causal validation is a class of methods for determining whether the causal model is correctly specified [9].

The primary goal of causal discovery and validation is to determine whether the generated model is consistent with the data. Relationships between variables must be discovered to generate the causal model. To achieve this, dependence between variables is measured using independence testing. Two random variables in a causal model are independent if a change in one of the variables does not affect the other when all other variables of the environment are kept constant. Measuring the CI between different sets of variables is an essential technique for both Causal Discovery and Validation [10]. Independence Testing has two main subcategories, unconditional and conditional independence testing [11]. Unconditional independence testing considers the" variable *X* is dependent on the variable *Y* directly" denoted by $X \not\perp\!\!\!\perp Y$ as the null hypothesis and tries to approximate the *p*-value for this hypothesis from the given data [12]. For conditional independence, consider a scenario: let *X*, *Y*, and *Z* denote sets of random variables, then the independence between *X* and *Y* given *Z* is denoted by $X \perp\!\!\!\perp Y / Z$. Both the conditional and unconditional independence tests are key in causal validation to know the relationships between variables. Generally, conditional independence testing is much more complicated and time-consuming than unconditional independence testing [12]. This is due to non-linearity and noise in the data and the "curse of dimensionality" for the variable *Z*. The test statistic for conditional independence is the distance between the estimated conditional densities $p\ (X|Y, Z)$ and $p(X|Y)$ [13].

A standard metric used to measure the dependence between two variables, which is the basis of independence and conditional independence testing, is the linear correlation between the two variables [14]. But as the name suggests it can only detect and map the linear relationship between the variables. A false negative detection for the correlation may indicate independence when the variables on verification may be dependent or vice versa [15]. The real world is rarely linear; thus, a better metric is required to map the correlation between variables accurately. The higher dimensions of the environment variables create a computational barrier for approximating the independence between the variables. The same problem is more relevant for conditional independence, where both the conditional variables and the target variable determine the dimensionality of the test and incur the "curse of dimensionality". A solution to the curse of dimensionality is explored in [16] which uses kernel feature maps (functions), to map the random variable from its original nonlinear space to a kernel space where linear operations can be performed on it.

The most useful kernel space is the Hilbert space, which is the complete vector space on the distance function, induced by the inner product, that defines the kernel space [17]. Two-dimensional and three-dimensional pictures can be used to reason about infinite-dimensional Hilbert spaces.

$$\langle y, x\rangle = \langle \underline{x, y}\rangle \tag{1}$$

$$\langle ax_1 + bx_2,\ y\rangle = a\langle x_{1,}y\rangle + b\langle x_{2,}y\rangle \tag{2}$$

Eq. (1) shows that the inner product is conjugate symmetric, i.e., for real-valued variables the complex conjugate is equal to the complex number. This implies:

$$\langle x_{,} x\rangle > 0 \,|\, x \neq 0$$

$$\langle x_{,} x\rangle = 0 \,|\, x = 0 \tag{3}$$

Eq. (2) describes that the inner product is linear in its first argument and Eq. (3) constrains the inner product to be positive definite. A special type of Hilbert space that allows scaling of vector space from its non-linear space to a linear space is the Reproducing Kernel Hilbert Space (RKHS). An RKHS forms the mathematical base for the RCoT algorithm. RCoT uses RKHS to translate vectors from nonlinear space to linear space. An RKHS is defined below.

Consider a Hilbert space *F* of functions from $X \rightarrow R$. Then *F* is a reproducing kernel Hilbert space if, for each $x \in X$, the Dirac evaluation operator δ_{x}*:* $F \rightarrow R$, which maps $f \in F$ to $f(x) \in R$, is a bounded linear functional. To each point $x \in X$, there corresponds an element $\phi(x) \in F$ such that $\phi(x), \phi(x')_{F} = k(x, x')$, where $k: X \times X \rightarrow R$ is a unique positive definite kernel [18]. The above definition will require that *F* be separable (it must have a complete orthonormal system). Such a reproducing kernel exists if and only if every evaluation functional is continuous [19]. The Hilbert Schmidt Independence Criterion (HSIC) uses the distance between the kernel embeddings of probability measures in the Reproducing Kernel Hilbert Space (RKHS). RKHS theory is normally described as a transform theory between RKHS and positive semidefinite functions, called kernels [20–22]. RKHS are precisely the space of functions where norm convergence implies pointwise convergence and are, consequently, relatively well behaved compared to other Hilbert spaces.

III. RCoT Algorithm

The Hilbert-Schmidt norm of the normalized conditional cross-covariance operator is used to show that this operator encodes the dependence structure of random variables. It proves in the limit of infinite data, under assumptions on the richness of the RKHS, that this measure has an explicit integral expression that depends only on the probability densities of the variables, despite being defined in terms of kernels [4].

The Kernel Conditional Independence Test (KCIT) is a widely used Conditional Independence test in the non-parametric setting, but KCIT scales cubically with sample size. A solution to the above limitation of KCIT is to use Random Fourier features to scale down the dimension of KCIT to simplify the computation [23]. This paper introduces the Randomized Conditional Independence Test (RCIT) and the Randomized Conditional Correlation Test (RCoT). RCIT explores the partial cross-covariance matrix of (*X*, *Y*), and RCoT explores the correlation of *X* and *Y* after subjecting the variable sets to the nonlinear transformations and then nonlinearly regressing out the effect of *Z*.

In practice, both proposed tests scale linearly with sample size and return accurate *p*-values much faster than KCIT, when tested on large sample size.

RCoT Algorithm uses Random Fourier Features to significantly improve the performance by approximating the results of the kernel. It further optimizes the performance by using a powerful approximation technique, i.e., the Lindsay Pilla Basaky method to determine whether the observed dependence is statistically significant or caused by random variation in the data [2].

Consider a random variable (*X*, *Y*) and RKHS H_x and H_y on *X* and *Y*, respectively, with measurable positive definite kernels k_x and k_Y.

$$E\,[k_x(X\,,X)]\ <\ inf,\quad E\,[k_x(Y\,,Y)]\ <\ inf \tag{4}$$

The cross-covariance operator *Y*, *X*: $H_x \rightarrow H_y$ is defined by the unique bounded operator that satisfies:

$$\langle g\,,\Sigma_{Y\,X}\rangle_{H_y}\ =\ Cov[f(X),g(Y)]\ (E[f(X),g(y)] - E[f(X)]\,E[\,g(Y)]) \tag{5}$$

It is known that the cross-covariance operator can be decomposed into the covariance of the marginals and the correlation; that is, there exists a unique bounded operator V_{YX} such that

$$\Sigma_{Y\,X}\ =\ \Sigma_{YY}^{1/2} V_{Y\,X}\,\Sigma_{Y\,X}^{1/2} \tag{6}$$

Consider another random variable *Z* on *Z* and RKHS (H_z; k_z). The normalized conditional cross covariance operator is defined as,

$$V_{Y\,X\,|\,Z}\ =\ V_{Y\,X}\ -\ V_{Y\,Z}\,V_{Z\,X} \tag{7}$$

$$V_{Y\,X\,|\,Z}\ =\ \Sigma_{YY}^{-1/2}\ (\Sigma_{Y\,X}\ -\ \Sigma_{Y\,X}\ \Sigma_{Z\,Z}^{-1}\ \Sigma_{Y\,X})\ \ \Sigma_{Y\,X}^{1/2} \tag{8}$$

Using the above equations, the $\Sigma_{Y\,X}$ operator can be used to determine the independence of *X* and *Y*, i.e., $\Sigma_{YX} = 0$ if and only if $X \perp\!\!\!\perp Y$.

After evaluating the *p*-value and the test statistic for the data, we must analyze the result to confirm whether it is significant or could have been caused by random chance. For this test, two approximation methods are used, Hall-Buckley-Eagleson (HBE) and Lindsay-Pilla-Basak (LPB) [17]. HBE is a precursor to LPB, so the latter is explained below. It is a moment-based approximation of random variable distributions using mixtures, and it is used to improve the efficiency of the RCoT algorithm. LPB uses moment methods to approximate a theoretical univariate distribution with a mixture of unknown distributions. The mixture used to approximate is a finite mixture of *n* Gamma cdfs. The LPB is a complicated procedure, the Section IV-B goes through the algorithm sequentially.

IV. Implementation

This section describes the first implementation of the RCoT Algorithm and LPB approximation in Python and reports the results of the same. Most of the Linear algebra equations are not present directly in Python, so numerical operations such as HBE and LPB [17] approximation techniques were implemented. The RCoT algorithm uses Random Fourier Features to accelerate the training of kernel machines by mapping the input data to randomized low-dimensional feature space and then applying linear methods, in this case, cross-correlation to reduce the computation time compared to traditional kernel-based independence testing methods. The output of the RCoT algorithm is the *p*-value for the null hypothesis, "the variables are dependent on each other". Dataset used for testing and the link to the implementation code are provided in Section IV-C.

A. *Pseudo Code for RCoT*

- Input: n datapoints of three random variables *X*, *Y*, *Z*.
- Verify if conditional variable *Z* exits in given data.
- If it does not exist run the Unconditional Independence Testing algorithm, i.e., Residual Independence Test (RIT).
- Slice the given data to extract the fixed number of datapoints for the Random Fourier Features (RFF) method.
- The output of the RFF algorithm is the input data mapped to a low-dimensional feature space in the form of a matrix(feat) for each of the variables.
- Normalize the feat matrix for each of the variables (f_x, f_y, f_z).

- Invert the normalized matrix of conditional variable (f_z) after converting it to positive definite.
- Find covariance of normalized matrix of X (f_x) with respect to normalized matrix of Z (f_z). Similarly for Z (f_z) and Y (f_y).
- Evaluate the test statistic for linear partial correlation using the values of $C_{x,y,z}$ (cross-correlation of x with respect to y given z).
- Find the eigenvalues of the Covariance matrix and apply Lindsay-Pilla-Basaky approximation with the eigenvalues as *coeff* vector and test statistic value for x to determine if the output of RCoT is statistically significant or it could have been generated by a random variation.
- The RCoT algorithm return Test statistic Sta, the *p*-value for the null hypothesis.

B. *Pseudo Code for RCoT*

- Input: *coeff* is the coefficient vector, *x* is a vector.
- If the length of *coeff* is less than 4 run the HBE approximation and return the *p*-value.
- Set p, which is the number of support points, value to 4 (more support points increase the accuracy and the computational intensity).
- Compute the weighted sum of first the 2p chi-squared moments and solve its determinant equation. Use bisection method to find a *lambdatilde_p*.
- Generate the delta matrices for each datapoint. Use the calculated *lambdatilde_p* to generate *Mp* which will be used to create matrix $\mathcal{S}$.
- Compute polynomial coefficients of the modified *Mp*, obtain the real roots of the polynomial and store it as a vector.
- Generate the Vandermonde matrix using the vector of roots.
- Calculate the linear combination of the I gamma cdfs using *lambdatildep* and *mui* as parameters. Return this value.

C. *Dataset and Testing*

Synthetic Data was generated to test the implementation of the algorithm described above. Data is generated using random probability distributions like uniform distributions, normal distributions, logistic distributions, exponential distributions, and gamma distributions.

Some exogenous variables are declared, and their values are generated using the linear combination of the above distributions from the NumPy. random library.

Then, the values of the endogenous variables are generated using linear and non-linear combinations of exogenous variables. The paper used 10 different types of models with different combinations of distributions to generate exogenous variables and different combinations of exogenous and endogenous variables to generate endogenous variables.

The data generated is used to test the RCoT algorithm and the results are compared with the results obtained from the Probspace algorithm. An example of the one model used in the data generations and its corresponding graph is shown in Fig. 2(a) and Fig. 2(b), respectively.

The model shown has one exogenous variable B and four endogenous variables, A, C, D, and E. B is a linear combination of the logistic function. logistic (0,1) is added to every variable to simulate noise. The model array stores the parents of every variable in the model. The varEquations array stores the mathematical equations (using Python APIs) that are used to generate datapoints.

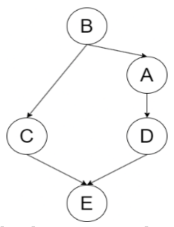
The data is generated based on the number of points we require to run. The example shown is simple with a few dependencies between the variables, but much more complex data with multilevel chain and fork dependencies were used for testing. The data generated is used to test the RCoT algorithm and the results are compared with the results obtained from the Probspace algorithm. The code for the implementation of the RCoT algorithm and the LPB approximation is available at https://github.com/mayank-agarwal-ln/RCoT.

Then tests were generated for the data. Tests generated for the above model are shown in Fig. 3. Some test dependency and others the independence between variables.

```
model =    [('B', []),
            ('A' , ['B']),
            ('C', ['B', 'A', 'D']),
            ('D', ['A']),
            ('E', ['C', 'D']),
            ]

varEquations = [
                 'B = 2 * logistic(2,3) + 0.1',
                 'A = 1 * B + logistic(0, 1)',
                 'D = .5 * A + logistic(0, 1)',
                 'C = 1.5 * B + logistic(0, 1)',
                 'E = 0.5 * C + 0.75 * D + logistic(0, 1)'
]
```

(a) Example of the model used to generate the data



(b) Graph of the data generated using the model

Figure 2. Example of the model used to generate the data and the corresponding graph.

```
# List of independent relationships
indeps = [('C', 'A', ['B']),
          ('B', 'D', ['A']),
          ('A', 'E', ['D']),
          ('B', 'E', ['C']),
          ]

# List of dependent relationships
deps = [('A', 'B'),
        ('D', 'A'),
        ('B', 'E')
        ('C', 'B', ['A', 'D']),
        ('E', 'C', ['D']),
        ('B', 'D', ['C'])
        ('C', 'D', ['E'])
        ]
```

Figure 3. Tests generated for the model shown in Fig. 2(a).

In the tests shown in Fig. 3, the first independence test shows the fork (common cause) rule which states that if *X* is the common cause of *Y* and *Z* and there is only one path between *Y* and *Z* then *Y* is conditionally independent of *Z* given *X*. The other tests show the chain rule, that if two variables *X* and *Y* are connected unidirectionally with one or more variables in the path, then *X* is conditionally independent of *Y* given *Z*, where *Z* is a subset of all variables in the path between *X* and *Y*.

In the list of dependents, most are direct dependents and multilevel dependents, except the last one which shows the common effect rule, which states that two exogenous variables *X* and *Y* are conditionally dependent given *Z* if there is a common effect of *Z* is the common effect of *X* and *Y*. The tests were run with Python 3.6 on a 64-bit Windows 10 machine with 8GB RAM and a 2.5 GHz Intel Core i5 processor. The results of the tests are shown in Fig. 4 are discussed in Section V.

V. Results

The following section compares the performance of the Probspace algorithm with the RCoT algorithm, a kernel-based conditional independence testing algorithm. Data used to test the algorithms were synthetically generated based on predefined causal models described in Section IV-C. The testcases are presented to the algorithm. The output of both algorithms is the dependency between the variables or conditional dependency when more than two variables are tested.

We have chosen to compare the time taken for the test to complete when the number of datapoints is varied to get an understanding of the efficiency of the algorithm. The accuracy of the algorithm is tested separately and compared to Probspace.

The algorithm also outputs the *p*-value for the test performed. This value is used to determine the statistical significance of the results produced. The paper considers the *p*-value threshold to be 0.05, which is common in the literature.

The total time is then recorded for the number of datapoints. In Fig. 2(b), the average time Probspace takes to complete the testcases is ≈ 6s and for RCoT the average time is ≈ 3s. The results show that the RCoT algorithm, on average, takes less time to compute the dependency between variables compared to the Probspace algorithm.

This makes the implemented Python version of the RCoT algorithm more useful in places where statistical dependence between variables is of importance. The graph below represents the results. There is a 50% reduction in the time taken for the completion of the test cases.

The accuracy of RCoT and Probspace are comparable, and on average are the same. The following tests can be performed only for forks, chains, or multilevel dependencies. RCoT matches Probspace's accuracy for all these tests. The paper tested both algorithms for multilevel dependencies. RCoT was able to detect up to 4 levels of direct dependency. Fig. 5 shows the testcase and Table I shows the results. The dependency values of 1 for the first four levels show that direct dependency is detected even with noise.

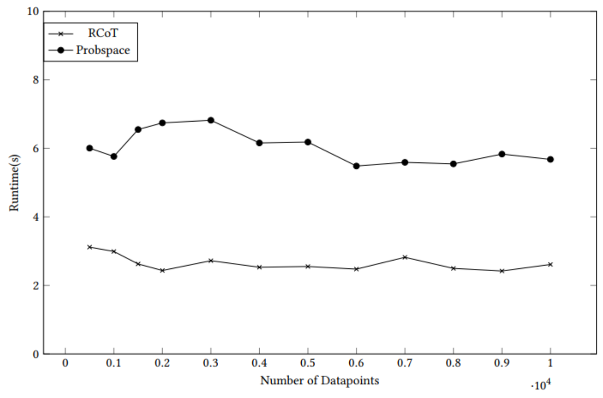


Figure 4. The runtime of the algorithm against the number of datapoints, for RCoT and Probspace.

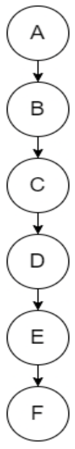


Figure 5. Testcase generated for multilevel dependency.

TABLE I. Dependency Values for Test Case

Test	Dependency Value
A-B	1
A-C	1
A-D	1
A-E	1
A-F	0.76

In any testcase, the conditioning set usually determines the time taken to complete the test case. The algorithms were tested with different dimensionalities of the conditioning set to quantify its effect on the completion time for the algorithm. The results are given below in Table II. There is a sharp rise in the time taken when the dimensionality is > 4. Further tests must be performed to the reason for this increase.

TABLE II. Time Taken for the Algorithm to Complete the Testcases for Different Dimensionality of Conditioning Set

Dimensionality	Time — Probspace (s)	Time — RCoT (s)
0	0.83797	0.82909
1	0.90799	0.86583
2	3.20132	3.13366
3	12.48197	11.76319
4	49.50360	44.512780
5	173.11749	161.20707
6	594.8133	560.20229

VI. Conclusion

Statistical methods, especially inferential statistics are useful for explaining the work of a machine learning model and finding the correlation between random variables much simpler. After comparing kernel and non-kernel-based methods for conditional independence testing, it was identified that the kernel-based methods produce more accurate results. The disadvantage of using KCIT is its exponential scaling with dataset size, which would make it time inefficient for large datasets. To resolve this RCoT algorithm was introduced, which uses the RKHS method to map data to a lower-dimensional space. RCoT further uses the LPB approximation technique to achieve better accuracy. To further test and improve the model, further implementations should test on real-world variables. The implementation of RCoT in Python showed better results in terms of time efficiency compared to the existing algorithm, Probspace.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

M. Agarwal and A. Kashyap conducted the research together and wrote the paper. They were assisted and adviced by G. Shobha, J. Shetty and R. Dev.

Acknowledgment

We would like to thank LexisNexis Risk Solution, Alpharetta, GA, USA, and R. V. College of Engineering, for their support and encouragement during this research. All authors approved the final version.

References

[1] J. Pearl, *Causality*, 2nd ed. Cambridge University Press, 2009.

[2] C. Li and X. Fan, "On nonparametric conditional independence tests for continuous variables," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 12, 2019.

[3] K. Baba, R. Shibata, and M. Sibuya, "Partial correlation and conditional correlation as measure of conditional independence," *Australian and New Zealand Journal of Statistics*, vol. 46, pp. 657–664, 2004.

[4] K. Zhang, J. Peters, D. Janzing, and B. Schölkopf, "Kernel-based conditional independence test and application in causal discovery," arXiv pre-print, arXiv:1202.3775, 2012, doi: 10.48550/arXiv.1202.3775

[5] J. P. Ryan, S. Ament, C. P. Gomes, and A. Damle, "The fast kernel transform," in *Proc. the 25th International Conference on Artificial Intelligence and Statistics*, 2021, pp. 11669–11690.

[6] J. Pearl, "Causal inference," in *Proc. the Workshop on Causality: Objectives and Assessment at NIPS 2008*, I. Guyon, D. Janzing, and B. Schölkopf, Eds. 2010, pp. 39–58.

[7] P. Judea, "The seven tools of causal inference, with reflections on machine learning," *Commun. ACM*, vol. 62, pp. 54–60, 2019.

[8] D. Malinsky and D. Danks, "Causal discovery algorithms: A practical guide," *Philosophy Compass*, vol. 13, no. 1, p. e12470, 2018, doi: 10.1111/phc3.12470

[9] R. H. Heck, T. J. Larsen, and G. A. Marcoulides, "Instructional leadership and school achievement: Validation of a causal model," *Educational Administration Quarterly*, vol. 26, no. 2, pp. 94–125, 1990.

[10] P. Spirtes, C. Glymour, and R. Scheines, *Causation, Prediction, and Search*, MIT Press, 1993.

[11] H. Zhang, S. Zhou, K. Zhang, and J. Guan, "Causal discovery using regression-based conditional independence tests," in *Proc. the Thirty-First AAAI Conference on Artificial Intelligence*, 2017, pp. 1250–1256.

[12] W. Bergsma, "Testing conditional independence for continuous random variables," in *Eurandom Report*, 2004.

[13] L. Su and H. White, "A nonparametric Hellinger metric test for conditional independence," *Econometric Theory*, vol. 24, no. 4, pp. 829–864, 2008.

[14] K. Fukumizu, A. Gretton, X. Sun, and B. Schölkopf, "Kernel measures of conditional dependence," in *Proc. the 20th International Conference on Neural Information Processing Systems*, 2007, pp. 489–496.

[15] A. Gretton, R. Herbrich, A. Smola, O. Bousquet, and B. Scholkopf, "Kernel methods for measuring independence," *Journal of Machine Learning Research*, vol. 6, pp. 2075–2129, January 2005.

[16] W. Rudin, *Real and Complex Analysis (Higher Mathematics Series)*, McGraw-Hill, 1987.

[17] B. Levitan, "Hilbert space," *Encyclopedia of Mathematics*, 2001.

[18] A. Gretton, O. Bousquet, A. Smola, and B. Schölkopf, "Measuring statistical dependence with Hilbert-Schmidt norms," in *Algorithmic Learning Theory*, S. Jain, H. U. Simon, and E. Tomita, Eds. Berlin, Heidelberg: Springer, 2005, pp. 63–77.

[19] J. H. Manton and P.-O. Amblard, "A primer on reproducing kernel Hilbert spaces," arXiv:1408.0952, 2015, doi: 10.48550/arXiv.1408.0952

[20] A. Gretton, K. Borgwardt, M. J. Rasch, B. Scholkopf, and A. J. Smola, "A kernel method for the two-sample problem," arXiv pre-print, arXiv:0805.2368, 2008, doi: 10.48550/arXiv.0805.2368

[21] D. D. Zhang, H. F. Lee, C. Wang, B. Li, Q. Pei, J. Zhang, and Y. An, "The causality analysis of climate change and large-scale human crisis," in *Proc. the National Academy of Sciences*, vol. 108, no. 42, pp. 17296–17301, 2011.

[22] A. Smola, A. Gretton, L. Song, and B. Schölkopf, "A Hilbert space embedding for distributions," in *Algorithmic Learning Theory*, M. Hutter, R. A. Servedio, and E. Takimoto, Eds. Berlin, Heidelberg: Springer, 2007, pp. 13–31.

[23] E. V. Strobl, V. Shyam, and Z. Kun, "Approximate kernel-based conditional independence tests for fast non-parametric causal discovery," *Journal of Causal Inference*, vol. 7, pp. 1–24, March 2019.

[24] B. Lindsay, R. Pilla, and P. Basak, "Moment-based approximations of distributions using mixtures: Theory and applications," *Annals of the Institute of Statistical Mathematics*, vol. 52, no. 2, pp. 215–230, 2000.

[25] C. Glymour, K. Zhang, and P. Spirtes, "Review of causal discovery methods based on graphical models," *Frontiers in Genetics*, vol. 10, p. 524, 2019.

[26] M. Petersen and M. Laan, "Causal models and learning from data integrating causal modeling and statistical estimation," *Epidemiology*, vol. 25, pp. 418–426, 2014.

[27] M. Hein and O. Bousquet, "Kernels, associated structures and generalizations," Tech. Rep. 127, Max Planck Institute for Biological Cybernetics, Tübingen, Germany, July 2004.

[28] Q. Zhang, S. Filippi, A. Gretton, and D. Sejdinovic, "Large-scale kernel methods for independence testing," *Statistics and Computing*, vol. 28, pp. 113–130, Jan 2017.

[29] P. Hall, "Chi squared approximations to the distribution of a sum of independent random variables," *The Annals of Probability*, vol. 11, no. 4, pp. 1028–1036, 1983.

[30] D. Bodenham and N. Adams, "A comparison of efficient approximations for a weighted sum of chi-squared random variables," *Statistics and Computing*, vol. 26, 2015.

[31] M. Buckley and G. Eagleson, "An approximation to the distribution of quadratic forms in normal random variables," *Australian Journal of Statistics*, vol. 30A, pp. 150–159, 2008.

[32] M. J. Anderson and P. Legendre, "An empirical comparison of permutation methods for tests of partial regression coefficients in a linear model," *Journal of Statistical Computation and Simulation*, vol. 62, no. 3, pp. 271–303, 1999.