`ccrvam`: a Python Package for Model-Free Exploratory Analysis of Multivariate Discrete Data with an Ordinal Response Variable

Dhyey Mavani

Amherst College

Submitted to the Department of Mathematics and Statistics
of Amherst College in partial fulfillment of the requirements
for the degree of Bachelor of Arts with honors.

Advisor(s):
Professor Shu-Min Liao

April 28, 2025

Table of contents

Abstract **v**

Acknowledgements **vi**

1 Introduction **1**

2 Copulas and Association Measures **7**
2.1 Unraveling the Notion of Dependence 7
2.2 Copulas as a Unified Framework for Dependence 11
2.3 Sklar's Theorem and Invariance Principle 13
2.4 Copulas for Continuous and Discrete Data 15
2.5 Checkerboard Copula 17
2.6 Copula-based Measures of Association and Estimation 20

3 Checkerboard Copula Regression, its Visualization and Association measure for Model-Free Regression Dependence Analysis of Multivariate Discrete Data **31**
3.1 Data 31
3.2 Checkerboard Copula and its Density 32
3.3 Checkerboard Copula Score 33
3.4 Checkerboard Copula Regression, Prediction and Visualization 35
3.5 Checkerboard Copula Regression Association Measure 41
3.6 Visualization Methods 46

4 Software (Package) Implementation and Testing **48**
4.1 Set-up and Example Data 48
4.2 Types of Input Data Supported 49
4.3 Checkerboard copula score (for ordinal response variable) 52
4.4 Checkerboard copula Regression (CCR) 53
4.5 CCR Predicted Category Visualization 55
4.6 CCR Prediction Uncertainty Evaluation Using Nonparametric Bootstrap Resampling . 57
4.7 (S)CCRAM Estimation 59
4.8 (S)CCRAM Uncertainty Evaluation Using Bootstrap Resampling 60
4.9 Statistical Significance Testing for (S)CCRAM Using Permutation Test 64
4.10 Software Architecture and Design Principles 67
4.11 Testing, Validation, and Performance Evaluation 70

4.12 User Documentation and Example Workflows 73

5 Real Data Analysis **74**
5.1 Dataset Overview 74
5.2 Data Preparation and Loading 75
5.3 Exploratory Data Analysis 79
5.4 Calculating Checkerboard Copula Scores (CCS) 80
5.5 Checkerboard Copula Regression (CCR) Analysis 81
5.6 Quantifying Association with (S)CCRAM 82
5.7 Uncertainty Quantification Using Bootstrap 84
5.8 Statistical Significant Testing Using Permutation Tests 87
5.9 Bootstrap Analysis for CCR Predictions 87
5.10 Discussion and Clinical Interpretation 90

6 Conclusion and Future Work **94**

References **98**

Appendices **103**

A Code availability **103**
A.1 Chapter 2 Code 103
A.2 Chapter 3 Code 107

Abstract

Understanding regression dependencies among discrete variables—especially in the presence of ordinal responses—poses a persistent challenge in exploratory data analysis (EDA). While classical EDA techniques and continuous copula models have proven effective for continuous data, they often fail to capture the structure and interpretability required for categorical datasets. This thesis begins by critically evaluating these traditional approaches and highlighting their limitations in discrete settings. Motivated by these gaps, we explore the model-free dependence measures proposed by Wei and Kim (2021) and further advanced by Liao et al. (2024), which leverages the checkerboard copula framework to robustly characterize regression relationships in multidimensional contingency tables with both ordinal and nominal variables.

To operationalize this method, we present a novel, modular, and scalable Python package, `ccrvam`, designed to support efficient large-scale analysis. The package integrates with established scientific libraries such as NumPy, Pandas, SciPy, and Matplotlib, while incorporating Pytest and Sphinx for testing and maintainability. Through extensive simulations and real-world case studies, we demonstrate that `ccrvam` offers a powerful and flexible toolset for uncovering complex dependence structures in categorical data. Our contributions provide both a theoretical exposition and a novel practical resource for researchers engaged in data-driven exploration of discrete regression phenomena.

Acknowledgements

I am incredibly thankful to everyone who made this thesis journey one I will never forget.

First, I owe so much to Professor Shu-Min Liao. From my first research experience at Amherst—developing R-Blocks—to taking on the challenge of building `ccrvam` based on her most recent work, she trusted me, guided me, and pushed me to grow. She also introduced me to Professor Daeyoung Kim, whose thoughtful feedback and encouragement over the past year helped shape this project. I am grateful to both of them for believing in me and for their time and energy in my development.

I also want to thank Professor Nicholas Horton, my statistics advisor, who has supported me throughout my time at Amherst. His mentorship, encouragement, and always-clear advice have meant a great deal to me. I am also thankful to Professors Amy Wagaman, Jun Ishii, Matteo Riondato, Katharine Correia, Pamela Matheson, and Brittney Bailey. Their classes—Advanced Data Analysis, Advanced Econometrics, Data Mining, Missing Data Analysis, Intermediate Statistics, and Data Science respectively—gave me the foundation to take on this work. Special thanks to Professor Bailey for creating the `.qmd` thesis template and helping with the formatting and Python integration—her help made this process smoother.

Lastly, to my family: Mom, Dad, my sister, grandparents—thank you for everything. Your belief in me and your sacrifices made this possible. And to my friends and peers at Amherst—thank you for the discussions, competitions, club events, pickup games, and all the little moments in between.

Chapter 1

Introduction

Exploratory Data Analysis (EDA), a concept first articulated by John Tukey in 1977 (Tukey 1977), forms an indispensable foundation for statistical analyses and modern data science workflows. Tukey underscored the significance of letting data guide hypothesis generation rather than relying exclusively on predefined models or confirmatory statistical tests. According to Tukey, exploring data should precede formal modeling efforts, as it reveals patterns, structures, outliers, and underlying assumptions that would otherwise remain hidden or overlooked (Tukey (1977); Buja et al. (2009); Gelman and Vehtari (2021)). The classic example of Anscombe's quartet (Anscombe 1973) vividly illustrates how datasets with identical summary statistics—means, variances, and correlation coefficients—can exhibit dramatically different relationships once explored visually, reinforcing the necessity of visual exploration in the preliminary stages of data analysis.

In practical terms, EDA employs various visualization techniques and statistical summaries designed to enhance intuitive understanding and detect anomalies, thereby ensuring the validity and robustness of subsequent analyses. (Tufte (1983); Donoho (2017)) Modern analytical frameworks, such as the Cross-Industry Standard Process for Data Mining (CRISP-DM) (Shearer 2000), explicitly incorporate

EDA to inform data preprocessing, feature selection, and model construction. Effective EDA is crucial for identifying relevant variables, understanding multivariate interactions, and preventing misleading conclusions, particularly in high-dimensional or complex datasets.

As the dimensionality and complexity of datasets increase, traditional EDA methods face significant challenges, particularly in effectively capturing complex dependence structures within multivariate categorical data, which is typically presented as a multi-dimensional contingency table. There have been efforts to extend continuous data visualization methods to the discrete case, and many graphical methods were developed specifically for categorical data. (Liao et al. 2024)

While visualization remains a powerful tool, statistical methodologies capable of systematically modeling and quantifying dependence become increasingly necessary as complements. Copula theory provides a promising approach for addressing these complexities. A copula is a mathematical function that combines marginal distributions of individual variables into a comprehensive joint distribution, enabling distinct and flexible modeling of dependence structures independently from marginal behaviors.(Joe 2014; Nelsen 2006; Sklar 1959) Copulas have found widespread application across diverse fields, notably finance, engineering, and environmental studies, particularly due to their ability to capture intricate and non-linear dependence patterns beyond traditional correlation measures.

Despite copulas' advantages in modeling continuous data, their application to discrete or categorical variables presents unique challenges. The copula representation is generally not unique for discrete data, complicating standard analytical procedures. Addressing these issues, the checkerboard copula has been proposed as a pragmatic and effective method for modeling discrete dependencies. This technique assigns uniform probabilities within discrete segments or 'blocks,' preserving the dependency structure intrinsic to categorical data while minimizing additional assumptions.(Wei

and Kim 2021) The checkerboard copula thus facilitates capturing complex dependencies that are otherwise challenging to quantify using conventional copula approaches.

One key aspect of statistical modeling that has consistently intrigued researchers across various disciplines—including social sciences, healthcare, economics, and behavioral research—is regression dependence. (Wei and Kim 2021) Specifically, regression dependence aims to quantify and interpret relationships where one variable, designated as the response, depends on a set of explanatory variables. Ordinal response variables present unique analytical challenges, necessitating methods specifically tailored to their inherent ordered structure.

Traditionally, assessing regression dependence in ordinal categorical data has relied primarily on parametric, model-based methods. Popular approaches include cumulative logit (proportional odds) models, cumulative link models with alternative links (probit, log-log, complementary log-log), adjacent-categories logit models, continuation ratio logit models, stereotype models, latent variable and association models, canonical correlation analyses, and correspondence analysis models. (Wei and Kim 2021) While these approaches offer robust frameworks for explanatory or predictive modeling, they impose specific structural assumptions about relationships within the data, potentially limiting their exploratory power.

Alternatively, non-model-based approaches have been developed to provide flexible association assessments without predefined functional forms. Commonly utilized methods include generalized Cochran-Mantel-Haenszel procedures, ordinal odds ratios, Kendall's tau and its variants, Goodman and Kruskal's gamma, Spearman's rank correlation, Somers' D, and Kendall's partial tau.(Wei and Kim 2021) Despite their widespread use, these traditional methods suffer from significant limitations. They primarily address pairwise associations, often treat variables symmetrically without clearly distinguishing between explanatory and response roles, and inadequately leverage the ordinal

information inherent within categorical data. Specifically, it is not straightforward to use non-model-based methods when our objective is to understand regression dependence in multivariate ordinal data with an ordinal response variable. (Wei and Kim 2021)

To address these limitations, recent methodological innovations by Wei and Kim (2021) have introduced the Checkerboard Copula Regression Association Measure (CCRAM) and its scaled counterpart, Scaled CCRAM (SCCRAM). The foundational tool of this method is the checkerboard (multilinear extension) copula (Genest et al. 2014; Schweizer and Sklar 1974), constructed through multilinear interpolation of discrete marginal distributions. Checkerboard copulas uniquely capture and represent the dependence structure among discrete ordinal variables, offering a robust basis for exploring multivariate contingency tables.(Denuit and Lambert 2005; Nešlehová 2007)

CCRAM is a non-parametric, model-free measure explicitly designed to quantify regression-type dependencies involving an ordinal response and multiple categorical explanatory variables. By leveraging checkerboard copula scores—which incorporate the informative ordinal rankings of variables—CCRAM provides a comprehensive, robust measure analogous to the R-squared statistic commonly used in linear regression but tailored specifically for categorical contexts. It flexibly accommodates high-dimensional data, providing interpretable summaries of dependencies within complex contingency tables. (Wei and Kim 2021)

The introduction of SCCRAM further enhances interpretability by scaling CCRAM values onto a standardized 0–1 interval. This scaling facilitates straightforward, intuitive comparisons of association strengths across diverse research contexts and datasets. Additionally, CCRAM and SCCRAM enable detailed decomposition of multivariate dependencies, allowing researchers to isolate and analyze contributions from individual variables or subsets of predictors—an advantage notably lacking in traditional non-model-based measures (Wei et al. 2023).

This thesis is based upon these recent methodological innovations highlighted in Wei and Kim (2021) and Liao et al. (2024), aims to understand these novel methods better and, for the first time, introduce `ccrvam`, a Python package encapsulating the end-to-end EDA workflow that allows researchers to run their experiments correctly and efficiently without worrying about implementational details of the methods all the time. In this work, we limit our focus to the development of EDA workflow for multivariate-categorical data with an ordinal response variable and a set of independent categorical (ordinal or nominal) variables in a multi-way contingency table.

The remainder of the thesis is structured as follows. Chapter 2 provides an extensive review of copula theory and key association measures, beginning with exploring the fundamental concepts of dependence and copula functions, emphasizing continuous data contexts. It includes a discussion of Sklar's theorem and the invariance principle, elaborates on copula modeling applicable to both continuous and discrete data, and covers essential copula-based association measures such as Kendall's tau and Spearman's rho, laying the theoretical foundation necessary for subsequent developments. Chapter 3 introduces Checkerboard Copula Regression (CCR), addressing the unique challenges posed by multivariate categorical data. It presents the theoretical formulation of checkerboard copulas, defines the checkerboard copula score, and illustrates the CCR methodology, predictive capabilities, visualization techniques, and the novel Checkerboard Copula Regression Association Measure (CCRAM). Chapter 4 focuses on the practical implementation and testing of CCR within `ccrvam`, the novel software package we developed, detailing the data requirements, software architecture, computational strategies, visualization tools, and rigorous validation processes. Chapter 5 provides a comprehensive empirical application of the proposed methods to real-world data, demonstrating the effectiveness of CCR in uncovering hidden dependence structures and improving analytical insights. Finally, Chapter 6 summarizes the key contributions, outlines the significance of integrating exploratory data analysis, copula theory, and novel association metrics, acknowledges current limitations, and suggests promis-

ing directions for future research, including methodological extensions and advanced visualization approaches.

Chapter 2

Copulas and Association Measures

2.1 Unraveling the Notion of Dependence

In this section, we aim to formalize concepts of dependence and association. We will use two bivariate random vectors to facilitate our understanding and visualize their relationships through Python code.

2.1.1 Motivating Example

Consider (X_1, X_2) and (Y_1, Y_2) to be bivariate random vectors, each consisting of 10000 independent data points, which are distributed with the joint distributions F_X and F_Y, respectively. Given these bivariate vectors, one might ask: How can I compare the relationship between (X_1, X_2) to the relationship between (Y_1, Y_2)? One of the measures that can help us compare and contrast these relationships is the Pearson correlation coefficient (commonly denoted as $\rho_{pearson}$). After preliminary calculations on a Python kernel, we can see that $\rho_{pearson}(X_1, X_2) \approx 0.802$, but on the other hand, $\rho_{pearson}(Y_1, Y_2) \approx 0.755$. From these measure values, the dependence between (X_1, X_2) is stronger than the dependence between (Y_1, Y_2). Although this agrees with our scatter plots in Figure 2.1, it is vital to note that $\rho_{pearson}$ only captures the linear dependence between the underlying random

variables at hand.

Upon observing the Figure 2.1 closely, we note that the marginal distributions of X_1 and X_2 are close to normal, unlike the marginals of Y_1 and Y_2. Moreover, we can see that the relationship between Y_1 and Y_2 is non-linear. This vast difference in marginals takes away our trust from the appropriateness of using $\rho_{pearson}$ as a measure to compare dependence between the data vectors at hand.

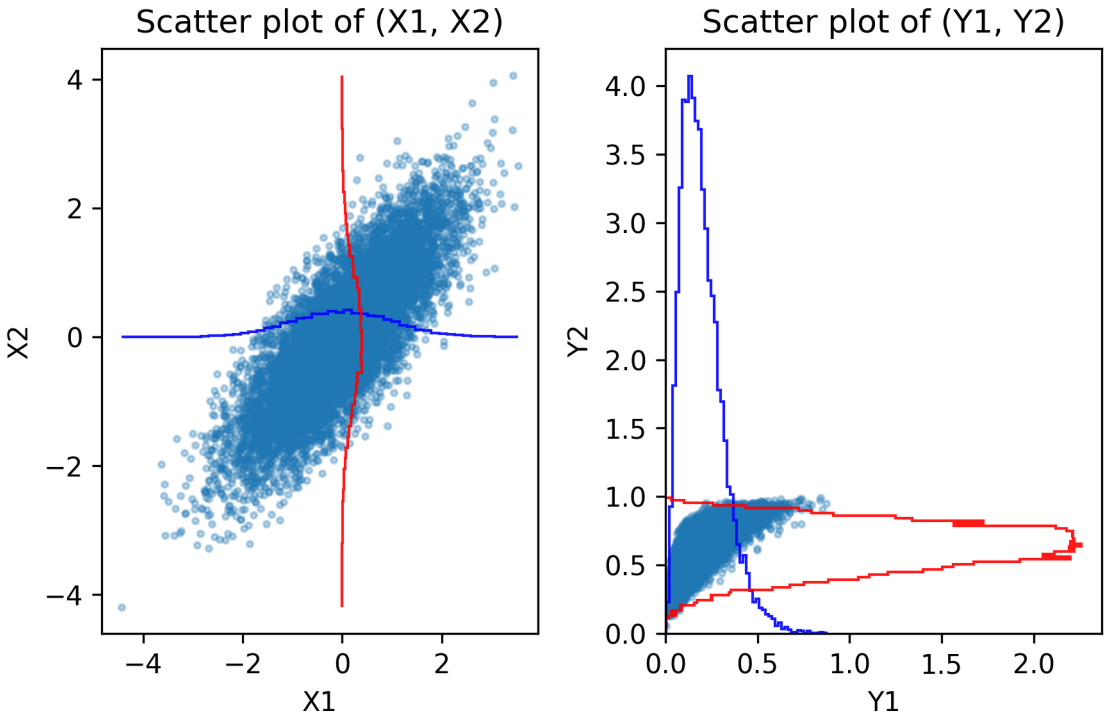


Figure 2.1: Scatter plots of 10000 independent observations of (X_1, X_2) and (Y_1, Y_2) with overlaid curves depicting respective marginal distributions.

Let us introduce a lemma that will help us transform the marginals so that the resulting marginals are more similar and try only to capture or extract the "dependence" components, allowing us to make fairer comparisons.

Lemma 2.1 (Probability Integral Transformation)**.** *(Hofert et al. 2018) Let F be a continuous distribution function and let $X \sim F$, then $F(X)$ is a standard uniform random variable, that is, $F(X) \sim U(0,1)$.*

Lemma 2.1 allows us to transform a continuous random variable into a random variable with standard

uniform distribution. So, by using this transformation, we can now convert our marginals X_1, X_2, Y_1, Y_2 individually to be distributed Uniform$(0, 1)$. Moreover, since the resulting marginals will all be the same type now, we can compare the dependence between random variables on fairer grounds.

For instance, if we know that $X_1 \sim N(0,1) = F_1$, $X_2 \sim N(0,1) = F_2$, $Y_1 \sim Gamma(3, 15) = G_1$, and $Y_2 \sim Beta(5, 3) = G_2$, where F_1, F_2, G_1, G_2 denote the distribution functions of the respective random variables. By Lemma 2.1, we can say that $F_1(X_1), F_2(X_2), G_1(Y_1)$, and $G_2(Y_2)$ are each distributed Uniform$(0, 1)$.

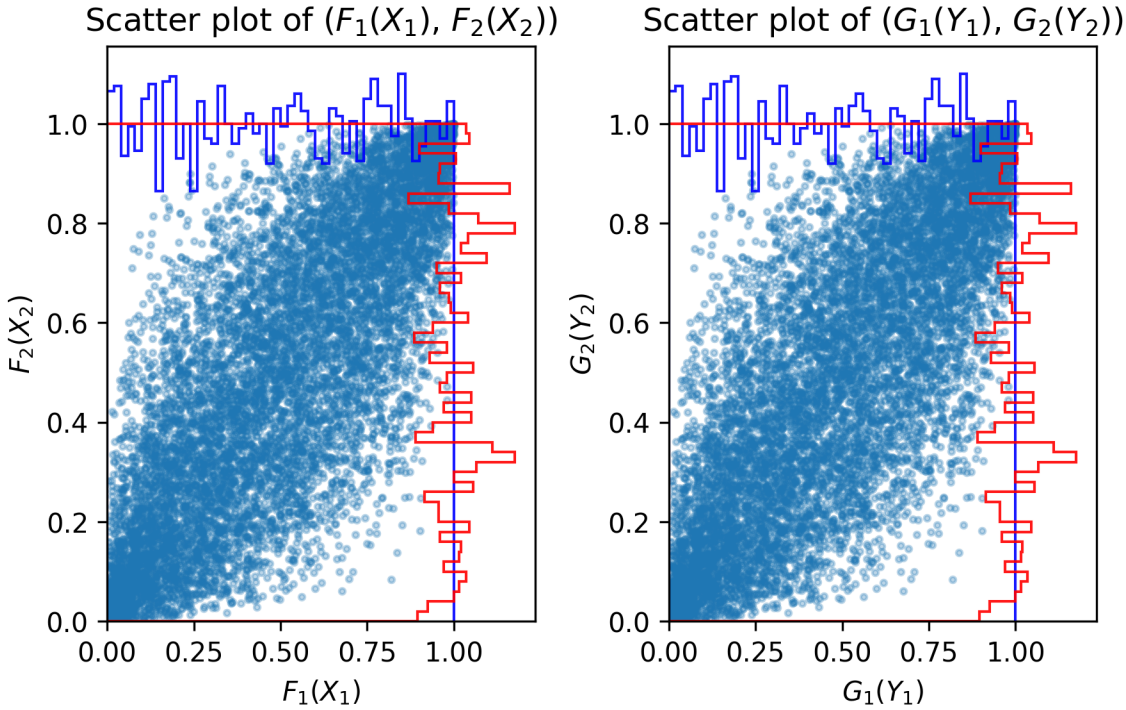


Figure 2.2: Scatter plots of 10000 independent observations of $(F_1(X_1), F_2(X_2))$ and $(G_1(Y_1), G_2(Y_2))$ with overlaid curves depicting respective marginal distributions.

Looking at the Figure 2.2, we can see that the transformed data vectors appear to be significantly similar. We can computationally verify this by quickly calculating the $\rho_{pearson}$ for $(F_1(X_1), F_2(X_2))$ and $(G_1(Y_1), G_2(Y_2))$, which turns out to be 0.788 for both data vector pairs, meaning that both have same dependence structures.

An alternative way to approach the problem (of comparing dependence of distinct pairs of marginals) is by transforming the marginals of (Y_1, Y_2) to be normal (same as marginals of (X_1, X_2)). As one can predict, in order to accomplish this transformation, we would need to "undo the current distributional mappings on (Y_1, Y_2)", which we can formally define as a generalized inverse as follows:

Definition 2.1 (Quantile Function)**.** (Hofert et al. 2018) $F^{\leftarrow}$ (Quantile Function) is defined as $F^{\leftarrow}(y) = \inf\{x \in \mathbb{R} | F(x) \geq y\}$, where $y \in [0, 1]$, and inf is the infimum of a set.

> ⚠ Warning
>
> The quantile function $F^{\leftarrow} = F^{-1}$ only when F is continuous and strictly increasing. Thus, it is important to note that, in other cases, the ordinary inverse F^{-1} need not exist. (Hofert et al. 2018)

With the above definition of $F^{\leftarrow}$, let us introduce a lemma from Hofert et al. (2018) that will help us perform the transformation to normal.

Lemma 2.2 (Quantile Transformation)**.** *(Hofert et al. 2018) Let $U \sim Unif(0, 1)$ and let F be any distribution function be a distribution function. Then $F^{\leftarrow}(U) \sim F$, that is, $F^{\leftarrow}(X)$ is distributed with density F.*

> ℹ Note
>
> Lemma 2.2 is valid for non-continuous densities F as well. (Hofert et al. 2018)

Let us start with the transformations where we left off in Figure 2.2 since we have uniform densities. Applying Lemma 2.2 on $G_1(Y_1)$ and $G_2(Y_2)$ using quantile functions $F_1^{\leftarrow} = F_1^{-1}$ and $F_2^{\leftarrow} = F_2^{-1}$ respectively gives us that $F_1^{-1}(G_1(Y_1)) \sim F_1$ and $F_2^{-1}(G_2(Y_2)) \sim F_2$.

Notice in Figure 2.3 that the resulting transformed distribution through this alternative method

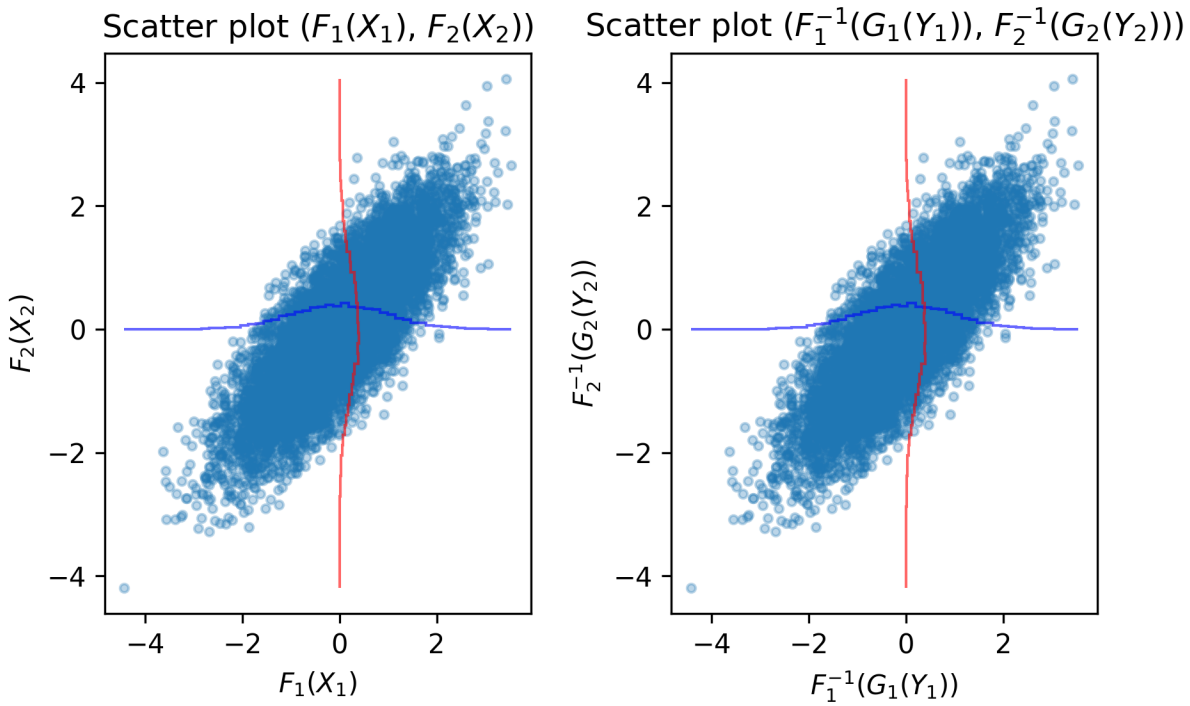


Figure 2.3: Scatter plots of 10000 independent observations of (X_1, X_2) and $(F_1^{-1}(G_1(Y_1)), F_2^{-1}(G_2(Y_2)))$ with overlaid curves depicting respective marginal distributions.

resembles that of (X_1, X_2). Hence, we can conclude that they have the same dependence. Furthermore, through a quick calculation, we can see that $\rho_{pearson}(F_1^{-1}(G_1(Y_1)), F_2^{-1}(G_2(Y_2))) = 0.802$, which is the same as the Pearson correlation coefficient between X_1 and X_2. This is the level of flexibility that a combination of transformations presented in Lemma 2.1 and Lemma 2.2 can lend us.

> **Note**
>
> "(X_1, X_2) and (Y_1, Y_2) have the same dependence" $\iff$ "(X_1, X_2) and (Y_1, Y_2) have the same copula" (Hofert et al. 2018)

2.2 Copulas as a Unified Framework for Dependence

Copulas are a class of multivariate distribution functions with Uniform$(0, 1)$ marginals. The motivating example in the previous section explains the usage of copulas as the structures capturing margin-

independent dependence between random variables.

> **Note**
>
> The choice of Uniform$(0,1)$ as a post-transformation margin for the data at hand is somewhat arbitrary, although it does simplify further results. One can use modifications of Lemma 2.1 and Lemma 2.2 to define copulas with respect to any margin of choice without affecting the conclusions about the dependence between the data at hand. (Hofert et al. 2018)

In order to understand copulas better, for now, let us restrict ourselves to the 2-D (2-dimensional) case. First, let us define a broader class of functions called subcopulas as a preliminary, which will help us mathematically define copulas as a special case. (Nelsen 2006)

Definition 2.2 (2-Dimensional Subcopula)**.** (Erdely 2017) A **two-dimensional subcopula** (2-subcopula) is a function $C^S : D_1 \times D_2 \longrightarrow [0,1]$, where $\{0,1\} \subseteq D_i \subseteq [0,1]$ for $i \in \{1,2\}$ with the following conditions satisfied:

- *Grounded:* $C^S(u,0) = 0 = C^S(0,v)$, $\forall u \in D_1, \forall v \in D_2$.
- *Marginal Consistency:* $\forall u \in D_1$ and $\forall v \in D_2$, $C^S(u,1) = u$ and $C^S(1,v) = v$.
- *2-increasing:* $\forall u_1, u_2 \in D_1$ and $\forall v_1, v_2 \in D_2$ such that $u_1 \leq u_2$ and $v_1 \leq v_2$, $C^S(u_1,v_1) - C^S(u_2,v_1) + C^S(u_2,v_2) - C^S(u_1,v_2) \geq 0$.

Definition 2.3 (2-Dimensional Copula)**.** (Erdely 2017) A **two-dimensional copula** (2-copula) is a function $C : [0,1] \times [0,1] \longrightarrow [0,1]$, with the following conditions satisfied:

- *Grounded:* $C(u,0) = 0 = C(0,v)$, $\forall u \in [0,1], \forall v \in [0,1]$.
- *Marginal Consistency:* $\forall u \in [0,1]$ and $\forall v \in [0,1]$, $C(u,1) = u$ and $C(1,v) = v$.
- *2-increasing:* $\forall u_1, u_2 \in [0,1]$ and $\forall v_1, v_2 \in [0,1]$ such that $u_1 \leq u_2$ and $v_1 \leq v_2$, $C(u_1,v_1) -$

$$C(u_2, v_1) + C(u_2, v_2) - C(u_1, v_2) \geq 0.$$

> **Note**
>
> A 2-D copula is just a 2-subcopula with a full unit square as domain ($D_1 = D_2 = [0, 1]$). Furthermore, copula and subcopula are the same within a domain with continuous variables. Later in this chapter, we will discuss why this does not hold when one of the variables is discrete.

In this expository chapter, we will mainly deal with 2-D copulas and subcopulas. However, the definitions above can be generalized to the n-D case with some notable exceptions detailed (with proofs) in section 2.10 of Nelsen (2006). Moreover, many different families of copulas bear peculiar properties and corresponding margins. We are not covering them in detail since that is not the focus of this work, and a comprehensive summary of many of these families can be found in chapter 3 of Hofert et al. (2018).

2.2.1 Fréchet-Hoeffding Bounds

For any distribution function, boundedness is always a desired property. In the case of copulas, we have a famous theorem that provides us with the upper and lower pointwise bounds.

Theorem 2.1 (Fréchet-Hoeffding Bounds)**.** *(Hofert et al. 2018) Given a 2-D copula C,* $W(u, v) = \max\{0, u + v - 1\} \leq C(u, v) \leq \min\{u, v\} = M(u, v)$*, where* $u, v \in [0, 1]$.

2.3 Sklar's Theorem and Invariance Principle

Theorem 2.2 by (Sklar 1959) is one of the seminal results in copula theory, which extended the applications of copulas and explained why copulas capture the dependence by relating the joint distributions to univariate margins.

Theorem 2.2 (Fréchet-Hoeffding Bounds)**.** *(Hofert et al. 2018)*

1. *Let H be a joint distribution function with univariate margins F and G. Then there exists a copula C such that $\forall x, y \in \mathbb{R}, H(x,y) = C(F(x), G(y))$. Furthermore, C is* ***unique*** *in the case when F, G are continuous; otherwise, in the general case, C is uniquely determined on $RanF \times RanG$, where $RanF, RanG$ denote the ranges of F, G respectively. That copula C is given by: $C(u,v) = H(F^{\leftarrow}(u), G^{\leftarrow}(u))$ such that $(u,v) \in RanF \times RanG$.*

2. *Conversely, H is defined as a 2-D distribution function with marginals F, G, if we are given copula C along with the univariate marginals F, G.*

The expository part of this thesis will mainly deal with two dimensions, but Theorem 2.2 above can be generalized to n-D case as detailed in section 2.10 of Nelsen (2006). Below, we include a few insights drawn from Hofert et al. (2018) that will be important to our ongoing discussion:

> **Note**
>
> Theorem 2.2 gives us an insight into the name copula and how it "couples" a joint distribution function to its marginal distributions. This coupling effect and two parts of Theorem 2.2 show us how to separate (or combine) multivariate dependence structure and univariate margins.

> **Spoiler Alert**
>
> In the case of continuous random variables, only one **unique** copula characterizes the multivariate dependence structure, which is very convenient for reasons we will discuss later in this chapter. This is not the case with discrete variables, which make the direct use of continuous copulas intractable.

> **Note**
>
> Theorem 2.2 can be used to verify the existence of a continuous distribution function H in the case of a multivariate dataset if and only if we are sure of the existence of corresponding continuous univariate marginals for each variable in the dataset.

2.3.1 The Invariance Principle

As we saw in the motivating example, the underlying dependence structure did not change over a specific type of transformation. This was very convenient for us; thus, it is a favorable property for a copula. This property is often formally referred to as "invariance," which we will formalize in the following theorem from (Hofert et al. 2018)

Theorem 2.3 (Invariance Principle)**.** *Let* $(X, Y) \sim H$ *with continuous margins* F, G *and copula* C*. If* T_X, T_Y *are* ***strictly increasing*** *transformations on* $RanX, RanY$*, respectively, then* $(T_X(X), T_Y(Y))$ *also has copula* C*.*

> **Note**
>
> Theorem 2.3 was implicitly in action during our analysis for the motivating example because the transformations that we used were of two kinds, namely, probability integral transformation and quantile transformation, and in both of the cases, we were dealing with continuous and **strictly increasing** mappings on the respective ranges of random variables.

2.4 Copulas for Continuous and Discrete Data

Up to this point, our discussion has centered on continuous random variables. Many of the results and definitions we have used rely on continuity, which ensures that the probability integral transform (PIT) maps each variable to a uniform distribution on $[0, 1]$. This property, in turn, guarantees the

uniqueness of the copula associated with a joint distribution via Sklar's theorem. In the continuous case, we have taken this uniqueness for granted.

However, real-world data are often **discrete**. When dealing with discrete random variables, the marginal distribution functions are not continuous, and the PIT no longer produces uniform random variables on the whole interval $[0, 1]$. Instead, we obtain what is known as a **subcopula**—a function defined only on a proper subset of $[0, 1]^2$, namely on the ranges of the marginal distributions.

> **i** Example: Bivariate Bernoulli Distribution
>
> *Imagine a bivariate distribution where each variable follows a Bernoulli law. In this setting, the only possible values for each variable are 0 and 1. The resulting subcopula is then defined on the set of points. Because this set is a proper subset of* $[0, 1]^2$*, the corresponding copula is not uniquely determined by the joint distribution of the variables.*

2.4.1 Unidentifiability Issue

Now, let us examine the unidentifiability problem in more detail. To illustrate the issue, consider the following adapted example from the two-dimensional case inspired by Geenens (2020). Suppose we have a subcopula C^S defined on a discrete domain, where $D_1 = \text{Ran}(F)$ and $D_2 = \text{Ran}(G)$ with the marginal distribution functions F and G, respectively. In the continuous case, a two-dimensional (sub)copula is defined on the entire unit square $[0, 1]^2$. By contrast, for discrete random variables, the subcopula C^S is only uniquely specified on the domain $D_1 \times D_2 = \text{Ran}(F) \times \text{Ran}(G)$.

To obtain a full copula C on $[0, 1]^2$, one must "fill in" the gaps—that is, extend the definition of C^S to those parts of the unit square not covered by $D_1 \times D_2$. Unfortunately, there are countless ways to perform this extension while still satisfying the fundamental properties required of a copula in its Definition 2.3. This leads to a **non-uniqueness** (or **unidentifiability**) issue, which complicates

the development and the application of copula-based models for discrete data. This unidentifiability has been examined in depth in the literature, such as Geenens (2020), and it calls into question the straightforward (direct) application of copula methods when at least one margin is discrete.

One of the ways to fill in the gaps is by performing a Distributional Transform, which adds random "noise" to each of the gaps in parent distribution as described by Rüschendorf (2009) and Faugeras (2017). After applying this, we can directly proceed to apply results from continuous copula modeling as we have smoothened out the discontinuities. Another method that accomplishes this goal is the multilinear extension of the subcopula, which leads to a copula commonly known as *Checkerboard Copula*.

2.5 Checkerboard Copula

In this section, we will define Checkerboard Copula (Genest et al. 2014), which is just a multilinearly interpolated copula on $[0,1]^d$ constructed from any subcopula C^S. Before diving into this definition, let us settle notations for some of our tools.

Let $\mathbf{X} = (X_1, \ldots, X_d)^T$ be a d-dimensional random vector with joint cumulative distribution function (c.d.f.) H, and arbitrary marginal c.d.f.s $F_1, \ldots, F_d$.

We know by Theorem 2.2, that there exists at least one copula $C : [0,1]^d \to [0,1]$ such that $H(x_1, \ldots, x_d) = C(F_1(x_1), \ldots, F_d(x_d)),$ where $x_1, \ldots, x_d \in \mathbb{R}$. As mentioned before, if the marginal c.d.f.s are continuous, then C is unique, and that $C = \mathbf{F}(\mathbf{X}) = (F_1(X_1), \ldots, F_d(X_d))$. However, in the case of discrete data, it is only uniquely determined on the domain $\Pi_{j=1}^{d} Ran(F_j)$.

Now, let's suppose that each X_j in $\mathbf{X}$ is an ordinal variable with finite number of categories I_j, which we can denote with numbers through a fixed ordering $\{x_1^j < \cdots < x_{i_j}^j < \cdots < x_{I_j}^j\}$. We need this ordering to represent our data in a d-way contingency table format that classifies our observations

with respect to the d-variables, each representing an axis and a fixed ordering, making sure that we can unambiguously locate the observations with a certain combination of categories within the contingency table.

Definition 2.4 (Joint p.m.f. in a d-way contingency table)**.** (Wei and Kim 2021) A **joint probability mass function (p.m.f.)** of $\mathbf{X}$ in the d-way contingency table is an array of size $\Pi_{j=1}^{d} I_j$ defined as:

$$P = \{p_{\mathbf{i}} = p_{i_1,\ldots,i_d} = \Pr(X_1 = x_{i_1}^1, \ldots, X_d = x_{i_d}^d) | i_j \in \{1, \ldots I_j\}, j \in \{1, \ldots, d\}, \mathbf{i} \in \Pi_{j=1}^{d} \mathbb{I}_j\}$$

where $\mathbb{I}_j = \{1, \ldots, I_j\}$ is the index set for X_j, $\mathbf{1} = (1, \ldots, 1)^T$ and $\mathbf{I} = (I_1, \ldots, I_d)^T$ denote index vectors of length d.

> **Note**
>
> As a consequence of the law of total probability, based on our notation, we also have that:
>
> $\sum P = \sum_{\mathbf{i}=\mathbf{1}}^{\mathbf{I}} p_{\mathbf{i}} = \sum_{i_1=1}^{I_1} \cdots \sum_{i_d=1}^{I_d} p_{i_1,\ldots,i_d} = 1$

Definition 2.5 (Marginal p.m.f. in a d-way contingency table)**.** (Wei and Kim 2021) A **marginal probability mass function (p.m.f.)** of i_j-th entry in X_j is:

$$p_{+i_j+} = \sum_{i_1=1}^{I_1} \cdots \sum_{i_{j-1}=1}^{I_{j-1}} \sum_{i_{j+1}=1}^{I_{j+1}} \cdots \sum_{i_d=1}^{I_d} p_{i_1,\ldots,i_d} = \sum_{\mathbf{i}_{-j}=\mathbf{1}_{-j}}^{\mathbf{I}_{-j}} p_{\mathbf{i}}$$

where $\mathbf{i}_{-j}$, $\mathbf{1}_{-j}$, and $\mathbf{I}_{-j}$ denote index vectors of $\mathbf{i}$, $\mathbf{1}$, and $\mathbf{I}$ with j-th entry omitted, respectively.

Thus, **marginal p.m.f.** for the $(d-1)$-dimensional random vector without X_j ($\mathbf{X}_{-j} = (X_1, \ldots, X_{j-1}, X_{j+1}, \ldots, X_d)^T$) is denoted by

$$p_{i_1,\ldots,+_j,\ldots,i_d} = \sum_{i_j=1}^{I_j} p_{i_1,\ldots,i_d}$$

Definition 2.6 (Conditional p.m.f. in a d-way contingency table)**.** (Wei and Kim 2021) A **conditional probability mass function (p.m.f.)** of X_j given $\mathbf{X}_{-j}$ is:

$$p_{i_j|\mathbf{i}_{-j}} = \frac{p_{i_1,\ldots,i_d}}{p_{i_1,\ldots,+_j,\ldots,i_d}}$$

Also, let us denote the (finite and discrete) range of the marginal distribution of X_j to be $D_j = \{u_0^j, \ldots, u_{i_j}^j, \ldots, u_{I_j}^j\}$. Then, $u_0^j = 0$, $u_{I_j}^j = 1$, and $u_{i_j}^j = \sum_{k_j=1}^{i_j} p_{+k_j+}$.

> **Note**
>
> If the superscript in $u_{i_j}^j$ can be trivially deduced from the subscript, then the superscript can be omitted for notational ease.

Then, by Theorem 2.2, the unique subcopula C^S associated with d-dimensional random vector $\mathbf{X}$ over $\prod_{j=1}^d D_j$ is given by $H(x_{i_1}^1, \ldots, x_{i_d}^d) = \sum_{k_1 \le i_1} \cdots \sum_{k_d \le i_d} p_{k_1,\ldots,k_d} = C^S(u_{i_1}^1, \ldots, u_{i_d}^d)$.

As mentioned before, any subcopula C^S on $\prod_{j=1}^d D_j$ can be extended to a copula C on $[0,1]^d$ by multilinear interpolation, Wei and Kim (2021) we define checkerboard copula as follows:

Definition 2.7 (Checkerboard Copula)**.** (Wei and Kim 2021)

Let C^s be a subcopula on $\prod_{j=1}^d D_j$ satisfying as defined above. For any $\mathbf{u} = (u_1, \ldots, u_d) \in [0,1]^d$, let u_j^ℓ and u_j^u be, respectively, the least and greatest elements of $\bar{D}_j$ (the closure of set D_j) satisfying $u_j^\ell \le u_j \le u_j^u$. Note that if u_j is in D_j, then $u_j^\ell = u_j^u$. Furthermore, for any $S \subseteq \{1, \ldots, d\}$, let

$$\lambda_j(u_j) = \begin{cases} \frac{u_j - u_j^\ell}{u_j^u - u_j^\ell}, & \text{if } u_j^\ell < u_j^u \\ 1, & \text{if } u_j^\ell = u_j^u \end{cases} \quad \text{and} \quad \lambda_S(u_1, \ldots, u_d) = \prod_{i \in S} \lambda_i(u_i) \prod_{i \notin S} (1 - \lambda_i(u_i)).$$

Then, the **checkerboard copula** C^+ of the ordinal random vector $\mathbf{X}$ is defined as

$$C^+(\mathbf{u}) = C^+(u_1, \ldots, u_d) = \sum_{S \subseteq \{1,\ldots,d\}} \lambda_S(u_1, \ldots, u_d) C^s(u_{s_1}, \ldots, u_{s_d}),$$

where $u_{s_j} = u_j^u$ if $j \in S$ and $u_{s_j} = u_j^\ell$ otherwise.

Additionally, by taking the derivatives of C^+ with respect to $u_1, \ldots, u_d$, the **checkerboard copula density function** is defined to be

$$c^+(\mathbf{u}) = c^+(u_1, \ldots, u_d) = \frac{p_{i_1,\ldots,i_d}}{\prod_{j=1}^d p_{+i_j+}}, \quad \text{where } u^j_{i_j-1} < u_j \leq u^j_{i_j}.$$

2.6 Copula-based Measures of Association and Estimation

Now that we have built an object (copula) that allows us to capture the multivariate dependence structure between variables, we would like to encode certain pieces of this information into a set of robust measures or metrics. We call these measures the **measures of association**. There are two types of measures of association: parametric and non-parametric. As discussed briefly for our motivating example, a common (parametric) measure of association is the Pearson correlation coefficient ($\rho_{pearson}$). Although it is efficient to calculate, it only captures linear dependence between the random data vectors. Let us discuss this metric in more detail, along with its limitations:

2.6.1 Pearson's Correlation Coefficient ($\rho_{pearson}$) & its Properties

Definition 2.8 (Pearson correlation coefficient)**.** Given a random vector (X, Y) with $Var(X) < \infty$ and $Var(Y) < \infty$, then:

$$\rho_{pearson}(X,Y) = \frac{Cov(X,Y)}{\sqrt{Var(X)}\sqrt{Var(Y)}}$$

, where covariance is defined as $Cov(X,Y) = \mathbb{E}((X - \mathbb{E}(X))(Y - \mathbb{E}(Y)))$, and the variance is defined as $Var(X) = \mathbb{E}((X - \mathbb{E}(X))^2)$.

Let's start by going over some commonly-used properties of $\rho_{pearson}$ as mentioned in Hofert et al. (2018):

1. $\rho_{pearson} \in [-1, 1]$
2. $|\rho_{pearson}(X,Y)| = 1$ if and only if $\exists a, b \in \mathbb{R}$, with $a \neq 0$ such that $Y = aX + b$ almost surely with $a < 0$ if and only if $\rho_{pearson}(X,Y) = -1$, and $a > 0$ if and only if $\rho_{pearson}(X,Y) = 1$. In both cases, X, Y are called *perfectly linearly dependent*
3. If X and Y ar independent, then $\rho_{pearson}(X,Y) = 0$.
4. $\rho_{pearson}$ is invariant under *strictly increasing linear* transformations.

2.6.2 Limitations of Pearson's Correlation Coefficient ($\rho_{pearson}$)

Although Pearson's correlation coefficient ρ_{pearson} is helpful in many cases, it only captures **linear dependence** and ignores non-linear relationships. Below, we summarize its key limitations along with illustrative examples.

- **Non-Existence of $\rho_{pearson}$:** Pearson's correlation does not exist for every random vector (X, Y), particularly when variances (or other higher order moments) are undefined.

Example: Heavy-Tailed Distributions

Consider two independent random variables X_1, X_2 drawn from a **Pareto(3)** distribution with $F(x) = 1 - x^{-3}, \quad x \geq 1$. Define $X = X_1$, and $Y = X_1^2$. The covariance is given by $Cov(X, Y) = Cov(X_1, X_1^2) = \mathbb{E}(X_1^3) - \mathbb{E}(X_1)\mathbb{E}(X_1^2)$. For Pareto(3), it is well-known (and can be easily proven) that $\mathbb{E}(X_1^3)$ **does not exist** (as the integral diverges). Since Pearson's formula relies on this moment, $\rho_{pearson}(X, Y)$ **does not exist**. On the other hand, we can observe that $Y = X^2$ shows a **perfect functional dependence** since Y can be represented as a deterministic (quadratic) function of X.

- **Non-Invariance Under Non-Linear Transformations:** $\rho_{pearson}$ is not necessarily invariant under all strictly increasing transformations on Ran$X$ or Ran$Y$.

Example: Logarithmic Transformation on $U(0, 1)$

Let $X \sim U(0, 1)$ and define $Y = \log(X)$. Pearson's correlation is: $\rho_{\text{pearson}}(X, Y) = \frac{\text{Cov}(X, \log X)}{\sigma_X \sigma_Y}$. Even though$Y = \log(X)$ is a **strictly increasing function**, $\rho_{pearson}$ changes under this transformation. Thus, Pearson's correlation is **not invariant** under (non-linear) monotonic transformations such as log in certain situations.

- **Uncorrelatedness Does Not Imply Independence:** $\rho_{pearson} = 0$ does NOT necessarily imply that (X, Y) are independent.

Example: Quadratic Transformation on $U(-1, 1)$

Let $X \sim U(-1, 1)$ and define: $Y = X^2$. We can compute: $\mathbb{E}[X] = 0, \quad \mathbb{E}[Y] = \mathbb{E}[X^2] = \frac{1}{3}$. Now, consider the covariance: $\text{Cov}(X, Y) = \mathbb{E}[XY] - \mathbb{E}[X]\mathbb{E}[Y] = \mathbb{E}[X^3] - (0)(\frac{1}{3})$. Since $\mathbb{E}[X^3] = 0$, we get $\text{Cov}(X, Y) = 0$. Thus, $\rho_{\text{pearson}}(X, Y) = 0$, but X **and** Y **are clearly dependent**, since knowing X exactly determines Y. This example demonstrates that a zero Pearson correlation

does **not** imply statistical independence.

- **Non-Uniqueness of the Joint Distribution Given Marginals and $\rho_{pearson}$:** The marginal distributions and the correlation coefficient do not uniquely determine the joint distribution.

> **i** Example: Bivariate Normal and Mixture Distributions
>
> Consider two bivariate distributions:
>
> 1. **Bivariate Normal Distribution**:
>
> $$(X_1, X_2) \sim N\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}\right).$$
>
> 2. **Bivariate Mixture Distribution** (Same Marginals, Different Dependence):
>
> $$X_1 \sim N(0,1), \quad X_2 = \begin{cases} X_1, & \text{with probability } 0.75, \\ -X_1, & \text{with probability } 0.25. \end{cases}$$
>
> Both cases yield: $\rho_{pearson}(X_1, X_2) = 0.5$.
>
> However, their **joint distributions are completely different**, meaning $\rho_{pearson}$ **does not uniquely determine dependence**.

- **Unattainability of Certain Correlations:** Given margins F_1, F_2, some $\rho_{pearson} \in [-1, 1]$ values cannot be attained by choosing any possible copula for (X_1, X_2). An example demonstrating this can be found in Hofert et al. (2018) p.46

In order to circumvent some of the limitations of the Pearson coefficient, we now consider rank-based correlation measures such as Spearman's Rho ($\rho_{Spearman}$) and Kendall's Tau ($\tau_{Kendall}$) as they only depend on the underlying copula C at least in the case of continuous random variables. Again, we

will discuss the peculiarities of the discrete case later in this chapter.

These rank-based measures are also known as **measures of concordance** (Hofert et al. 2018). We first define *concordance* to better understand this. Consider two points in $\mathbb{R}^2$, (x_1, y_1) and (x_2, y_2). These points are defined as concordant if $(x_1 - x_2)(y_1 - y_2) > 0$ and discordant if $(x_1 - x_2)(y_1 - y_2) < 0$.

2.6.3 Kendall's Tau

Definition 2.9 (Kendall's Tau)**.** Given a bivariate random vector (X_1, X_2) with continuous marginals F_1 and F_2, let's define (X_1', X_2') as an independent copy of (X_1, X_2). Then, the population version of Kendall's tau is defined by:

$$\tau_{kendall}(X_1, X_2) = \mathbb{E}(\text{sign}((X_1 - X_1')(X_2 - X_2')))$$

Here, $\text{sign}(x)$ is the sign function defined in a piecewise manner as follows:

$$\text{sign}(x) = \begin{cases} -1, & \text{if } x < 0, \\ 0, & \text{if } x = 0, \\ 1, & \text{if } x > 0. \end{cases}$$

Using the above-mentioned notion of concordance, definition of an expected value, and Definition 2.9, we can equivalently define Kendall's Tau as $\tau_{kendall} = (1)\mathbb{P}((X_1 - X_1')(X_2 - X_2') > 0) + (0)\mathbb{P}((X_1 - X_1')(X_2 - X_2') = 0) + (-1)\mathbb{P}((X_1 - X_1')(X_2 - X_2') < 0) = \mathbb{P}((X_1 - X_1')(X_2 - X_2') > 0) - \mathbb{P}((X_1 - X_1')(X_2 - X_2') < 0)$, since in the case of continuous distributions, probability at any given point is 0, specifically $\mathbb{P}((X_1 - X_1')(X_2 - X_2') = 0) = 0$.

As mentioned in Hofert et al. (2018) p.53, we can represent $\tau_{kendall}$ in terms of an underlying copula

C as $\tau_{kendall}(C) = 4\int_{[0,1]^2} C(u,v)d(C(u,v)) - 1$.

2.6.4 Spearman's Rho

Definition 2.10 (Spearman's Rho)**.** Given a bivariate random vector (X_1, X_2) with continuous marginals F_1 and F_2, then the population version of Spearman's rho is defined by:

$$\rho_{spearman}(X_1, X_2) = \rho_{pearson}(F_1(X_1), F_2(X_2))$$

We can observe that Spearman's rho is only Pearson's correlation coefficient of the transformed variables obtained after performing the Probability Integral Transformation defined earlier in Lemma 2.1.

As mentioned in Hofert et al. (2018) p.53, we can represent $\rho_{spearman}$ in terms of an underlying copula C as $\rho_{spearman}(C) = 12\int_{[0,1]^2} C(u,v)d((u,v)) - 3$.

> **Note:**
>
> $\tau_{kendall}$ and $\rho_{spearman}$ both overcome the significant limitations of $\rho_{pearson}$ with the following properties as summarized in Hofert et al. (2018):
>
> - These measures always exist and are invariant under all (not just linear) strictly increasing transformations
> - These measures attain all values in $[-1, 1]$, and they specifically attain -1 and 1 when the copula C attains the Fréchet-Hoeffding bounds W and M as defined in Theorem 2.1

So far, we have seen that in the continuous case, we have association measures that are already model-free and margin-free, but this is not the case in discrete situations, mainly because the representations of Spearman's rho and Kendall's tau in terms of copula C are not well-defined due to non-uniqueness of copula.

Significant research has been done on developing model-free and margin-free measures for the discrete case. In the following sub-section, we will discuss some measures for 2-dimensional discrete data from Denuit and Lambert (2005), Genest et al. (2014), Genest et al. (2017), and Nešlehová (2007).

2.6.5 Checkerboard Copula-based Association Measures for 2-Dimensional Data

Before understanding the 2-D modified versions of Kendall's tau and Spearman's rho that overcome the limitations induced by having "gaps" in the discrete case, let us define a simpler notation for the checkerboard copula in 2-D.

For a bivariate random vector (X_1, X_2) with discrete marginals F_{X_1} and F_{X_2}, we can construct the checkerboard copula C^+ through the following process:

1. Start with the empirical copula C defined on the range of $(F_{X_1}(X_1), F_{X_2}(X_2))$
2. Extend this copula to the entire unit square by creating a piecewise constant function over rectangles defined by the discontinuity points of the marginals
3. The resulting in a (visualizable) "checkerboard" pattern, which gives the approach its name

Mathematically, the resulting 2-D checkerboard copula can be represented as:

$$C^+(u, v) = \sum_i \sum_j C(a_i, b_j) \cdot \mathbf{1}_{(a_i, a_{i+1}] \times (b_j, b_{j+1}]}(u, v)$$

where a_i and b_j are the jump points of the marginal distributions, and $\mathbf{1}$ is the indicator function.

Checkerboard Version of Kendall's Tau

Denuit and Lambert (2005) and Nešlehová (2007) have shown that a modified version of Kendall's tau based on the checkerboard copula can be defined as:

$$\tau^+(X_1, X_2) = 4 \int_{[0,1]^2} C^+(u, v) dC^+(u, v) - 1$$

This measure maintains many desirable properties of the continuous case tau while being properly normalized for discrete data.

Checkerboard Version of Spearman's Rho

Similarly, a checkerboard version of Spearman's rho can be defined as:

$$\rho^+(X_1, X_2) = 12 \int_{[0,1]^2} C^+(u, v) du dv - 3$$

Strengths and Limitations of 2-D Checkerboard Approaches

The checkerboard-based association measures for discrete data represent a significant advancement in the field of dependence modeling. By addressing the "gap problem" inherent in discrete distributions, these measures provide more accurate and reliable quantification of dependence structures compared to their classical counterparts.

Empirical studies by Genest et al. (2017) demonstrate that the checkerboard versions consistently outperform traditional measures when applied to discrete data with varying degrees of ties. Their simulation results showed that τ^+ and ρ^+ maintain the proper type I error rates in hypothesis testing scenarios. In contrast, the traditional measures can be overly conservative or, in some instances,

anti-conservative.

Furthermore, Nešlehová (2007) established the asymptotic properties of these estimators, proving their consistency and deriving their limiting distributions under the null hypothesis of independence. This theoretical foundation enhances the statistical validity of inference procedures based on these measures.

The checkerboard-based measures mentioned above offer several important advantages:

1. **Proper Normalization**: Unlike traditional dependence measures, checkerboard-based measures are inherently bounded between -1 and 1. This boundedness simplifies comparison and interpretation, clearly indicating the direction and strength of dependence. The extreme values (-1 or 1) are attained explicitly under conditions of perfect monotone dependence, making them particularly useful for identifying maximal dependence scenarios accurately.

2. **Invariance**: A critical strength of checkerboard-based measures is their invariance under strictly increasing transformations of the marginal distributions. This means that these measures remain unaffected by transformations such as scaling or monotone data reshaping. Consequently, analyses based on these measures are robust, facilitating comparisons across datasets and ensuring reliability regardless of the chosen scale.

3. **Interpretability**: Checkerboard-based dependence measures consistently maintain interpretability irrespective of data discreteness. Traditional measures often struggle to offer clear interpretations when data exhibits discrete characteristics, leading to ambiguity. In contrast, checkerboard-based approaches offer intuitive interpretations, ensuring clarity on whether the data are discrete, continuous, or mixed, thus providing valuable insights into diverse practical applications.

4. **Compatibility**: An essential advantage of checkerboard-based measures is their compatibil-

ity with continuous marginal distributions. When applied to purely continuous data, these checkerboard-based measures naturally simplify the standard, widely used dependence measures. This property ensures seamless integration of these methods within existing analytical frameworks and allows for straightforward comparisons between traditional and checkerboard-based results.

Additionally, when implementing these checkerboard-based association measures, it is important to note that computational complexity increases with the number of distinct values in each marginal.

However, it is important to note that despite these advancements, the measures discussed thus far are primarily designed for bivariate (2-dimensional) data. In many real-world applications across fields such as bioinformatics, econometrics, and multivariate time series analysis, we encounter high-dimensional discrete data where pairwise association measures may not capture the full complexity of the dependence structure.

2.6.6 Looking Forward: Beyond Bivariate Measures

Although these 2-dimensional checkerboard approaches represent significant progress in the field, modern data analysis increasingly demands tools that can handle multivariate discrete data effectively. In Chapter Chapter 3, we will review Wei and Kim (2021)'s groundbreaking work on valid association measures for n-dimensional discrete data. Their framework extends the checkerboard concept to higher dimensions while preserving the desirable properties of margin freedom and model independence.

Chapter Chapter 4 will then present our contribution to the field: the first-ever Python package implementation of these n-dimensional measures. By making these advanced statistical tools accessible to practitioners through an open-source, well-documented, and computationally efficient Python

package, we aim to bridge the gap between theoretical advances and practical applications.

This implementation enables researchers across disciplines to analyze complex discrete multivariate data without the limitations imposed by traditional correlation measures or the need for parametric assumptions. The package includes comprehensive visualization tools, statistical testing procedures, and integration capabilities with popular data science frameworks.

By developing this implementation, we aim to facilitate wider adoption of these robust association measures in fields ranging from genomics to social network analysis, where discrete multivariate data is abundant but appropriate analysis tools have been limited.

Chapter 3

Checkerboard Copula Regression, its Visualization and Association measure for Model-Free Regression Dependence Analysis of Multivariate Discrete Data

In this chapter, we will review and combine concepts from Wei and Kim (2021) and Liao et al. (2024) to ultimately define regression based on checkerboard copula, along with some visualization and association measures for model-free regression dependence analysis of multivariate discrete data.

3.1 Data

Before diving into the details of the method and some prerequisites, let us understand the type of data of interest. The methods described in this chapter apply to multivariate categorical data in the form of a multi-dimensional contingency table with an ordinal response variable and a set of categorical (nominal/ordinal) predictors.

3.1.1 2-D Example

(Adapted from Wei and Kim 2021)

Consider a dataset that contains two ordinal variables: the dose of a treatment drug for acute migraine (X_1) with $I_1 = 5$ categories $((x_1^1, x_2^1, x_3^1, x_4^1, x_5^1) =$ (very low, low, medium, high, very high)), and the severity of migraine pain recorded after treatment (X_2) with $I_2 = 3$ categories $((x_1^2, x_2^2, x_3^2) =$ (mild, moderate, severe)).

Table 3.1: Joint p.m.f of X_1 and X_2, $P = \{p_{i_1 i_2}\}$.

$X_1 \backslash X_2$	x_1^2	x_2^2	x_3^2
x_1^1	0	0	2/8
x_2^1	0	1/8	0
x_3^1	2/8	0	0
x_4^1	0	1/8	0
x_5^1	0	0	2/8

This Example has been carefully constructed so that X_2 has a quadratic relationship with X_1 (as the level of X_2 decreases, the level of X_1 increases). We can also observe that X_2 is a function of X_1 with probability 1, but not the other way around (for a given category of X_1, there is one and only one category of X_2 whose corresponding joint probability is non-zero).

3.2 Checkerboard Copula and its Density

Recall that the checkerboard copula and the corresponding copula density are defined in Chapter Chapter 2's Definition 2.7. In order to understand it better, let's continue building upon the example data above by defining the copula and its corresponding density.

3.2.1 2-D Example (continued...)

(Adapted from Wei and Kim 2021)

Upon application of Definition 2.5, we can see that the marginal p.m.f.s of X_1 and X_2 are $p_{i_1+} \in \{2/8, 1/8, 2/8, 1/8, 2/8\}$ and $p_{+i_2} \in \{2/8, 2/8, 4/8\}$, respectively. Furthermore, we can see that the ranges of the marginal c.d.f.s of X_1 and X_2 are $D_1 = \{u_0^1, u_1^1, u_2^1, u_3^1, u_4^1, u_5^1\} = \{0, 2/8, 3/8, 5/8, 6/8, 1\}$ and $D_2 = \{u_0^2, u_1^2, u_2^2, u_3^2\} = \{0, 2/8, 4/8, 1\}$, respectively. As per Definition 2.7, we can visualize checkerboard copula density of X_1 and X_2 in Figure 3.1.

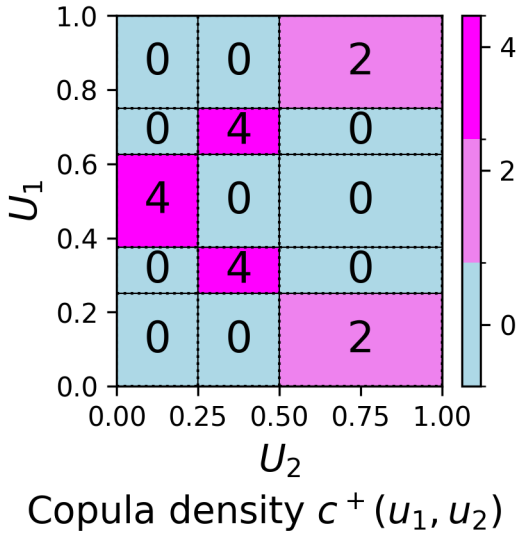


Figure 3.1: Checkerboard Copula Density Visualization for 2-D Example (This figure's styling is designed to reproduce the work presented by Wei and Kim, 2021 in supplementary materials)

3.3 Checkerboard Copula Score

Ordinal variables contain categories with natural ordering but unknown distances between them. We can leverage this inherent ordering information when analyzing associations in ordinal contingency tables. To achieve this, Wei and Kim (2021) introduced checkerboard copula scores derived from the checkerboard copula.

As established in Definition 2.7, the checkerboard copula represents a smoothed version of the subcopula associated with ordinal random vector X. It distributed probability mass uniformly across d-dimensional hyperrectangles in $[0,1]^d$, specifically within intervals $[u^j_{i_j-1}, u^j_{i_j}]$, where $u^j_{i_j}$ is defined by the marginal cumulative distribution functions. Furthermore, we can define a transformation of X_j via U_j as $S_j = \mathbb{E}[U_j|X_j]$, where $j \in \{1 \dots d\}$. Note that here, S_j is an ordinal random variable with numerical support values $\{s^j_1, \dots, s^j_{i_j}, \dots, s^j_{I_j}\}$, where $s^j_{i_j} = (u^j_{i_j-1} + u^j_{i_j})/2$.

Definition 3.1 (Checkerboard Copula Scores (CCS))**.** (Wei and Kim 2021) The **checkerboard copula scores (CCS)** of ordinal variable X_j are $\{s^j_1, \dots, s^j_{i_j}, \dots, s^j_{I_j}\}$, where $s^j_{i_j} = (u^j_{i_j-1} + u^j_{i_j})/2$ for $i_j \in \{1, \dots, I_j\}$ and $u^j_{i_j}$ as defined in Section 2.5. In other words, CCS is a set of the average of the marginal distributions evaluated at every two consecutive categories of X_j.

These scores have several interesting properties, as proven by Wei and Kim (2021). One of our interests is the formula for the mean and variance of the support vector S_j.

Lemma 3.1 (Mean and Variance of Support Vector (S_j))**.** *(Wei and Kim 2021) The probability-weighted mean (or expected value) of S_j is $\mu_{S_j} = 0.5$. The variance of S_j is $\sigma^2_{S_j} = \frac{1}{4}\sum_{i_j=1}^{I_j} u^j_{i_j-1} u^j_{i_j} p_{+i_j+}$.*

3.3.1 2-D Example (continued...)

(Adapted from Wei and Kim 2021)

Upon application of Definition 3.1, we obtain the checkerboard copula scores of X_1 and X_2 as $(2/16, 5/16, 8/16, 11/16, 14/16)$ and $(2/16, 6/16, 12/16)$ respectively. Furthermore, by Lemma 3.1, we can say that the $(\mu_{S_j}, \sigma^2_{S_j})$ of S_1 and S_2 are $(0.5, 81/1024)$ and $(0.5, 9/128)$ respectively.

3.3.2 Empirical Estimation of CCS

Since we are dealing with discrete count data, at times, we generally do not have access to the joint probability matrix. Instead, we have to base our analysis on the counts of observations with various categorical combinations of variables of interest. Thus, below, we establish the missing link between count data and pre-defined distributions as introduced by Wei and Kim (2021).

Let $\{n_{i_1,\ldots,i_d}\}, i_j \in \{1,\ldots,I_j\}, j \in \{1,\ldots,d\}$, denote counts in a d-way contingency table obtained by classifying $n = \sum_{i_1=1}^{I_1} \cdots \sum_{i_d=1}^{I_d} n_{i_1,\ldots,i_d}$ observations (or **cases**) into categories of d variables, $X_1,\ldots,X_d$.

Let's define marginal sums of i_j-th category in X_j as $n_{+i_j+} = \sum_{i_1=1}^{I_1} \cdots \sum_{i_{j-1}=1}^{I_{j-1}} \sum_{i_{j+1}=1}^{I_{j+1}} \cdots \sum_{i_d=1}^{I_d} n_{i_1,\ldots,i_d}$, and $(d-1)$-variate marginal frequencies of $\mathbf{X}_{-j}$ as $n_{i_1,\ldots,+_j,\ldots,i_d} = \sum_{i_j=1}^{I_j} n_{i_1,\ldots,i_d}$.

In terms of these, we can define estimators for the probabilities as follows:

$$\hat{p}_{i_1,\ldots,i_d} = \frac{n_{i_1,\ldots,i_d}}{n}, \hat{p}_{+i_j+} = \frac{n_{+i_j+}}{n}, \hat{p}_{i_1,\ldots,+_j,\ldots,i_d} = \frac{n_{i_1,\ldots,+_j,\ldots,i_d}}{n}, \hat{p}_{i_j|\mathbf{i}_{-j}} = \frac{\hat{p}_{i_1,\ldots,i_d}}{\hat{p}_{i_1,\ldots,+_j,\ldots,i_d}}$$

Moreover, the range of marginal c.d.f. of X_j is estimated by $[\hat{u}_0^j, \ldots, \hat{u}_{i_j}^j, \ldots, \hat{u}_{I_j}^j]$ with $\hat{u}_0^j = 0$ and $u_{i_j}^j = \sum_{k_j=1}^{i_j} \hat{p}_{+k_j+}$.

Using above-established pre-requisites, Definition 3.1, and Lemma 3.1 we can estimate the checkerboard copula scores $\hat{s}_1^j, \ldots, \hat{s}_{I_j}^j$ with $\hat{s}_{i_j}^j = (\hat{u}_{i_j-1}^j + \hat{u}_{i_j}^j)/2$ and $\hat{\sigma}_{\hat{S}_j}^2 = \sum_{i_j=1}^{I_j} \hat{u}_{i_j-1}^j \hat{u}_{i_j}^j \hat{p}_{+i_j+}/4$.

3.4 Checkerboard Copula Regression, Prediction and Visualization

Let $\mathbf{U}$ be a uniform random vector on $[0,1]^d$ associated with the checkerboard copula C^+ for a d-way ordinal contingency table.

Definition 3.2 ($(d-1)$-Marginal Density)**.** (Wei and Kim 2021) The $(d-1)$**-marginal density** for

$\mathbf{U}_{-j} = (U_1, \ldots, U_{j-1}, U_{j+1}, \ldots, U_d)^T$ is defined as

$$c^+(\mathbf{u}_{-j}) = \frac{p_{i_1,\ldots,+_j,\ldots,i_d}}{\prod_{k=1,k\neq j}^{d} p_{+i_k+}}$$

where $\mathbf{u}_{-j} = (u_1, \ldots, u_{j-1}, u_{j+1}, \ldots, u_d)^T$ in $[0,1]^{d-1}$ and $u^k_{i_k-1} < u_k < u^k_{i_k}$. Here, $k \in \{1, \ldots, j-1, j+1, \ldots, d\}$, $j \in \{1, \ldots, d\}$, $p_{i_1,\ldots,+_j,\ldots,i_d}$, and p_{+i_k+} as in Definition 2.5.

Definition 3.3 (Conditional Density of U_j given $\mathbf{U}_{-j}$)**.** (Wei and Kim 2021) The **conditional density of U_j given $\mathbf{U}_{-j}$**, where $\mathbf{U}_{-j} = (U_1, \ldots, U_{j-1}, U_{j+1}, \ldots, U_d)^T$ is defined as

$$c^+(u_j|\mathbf{u}_{-j}) = \frac{c^+(\mathbf{u})}{c^+(\mathbf{u}_{-j})} = \frac{p_{i_j|\mathbf{i}_{-j}}}{p_{+i_j+}}$$

where $\mathbf{u}_{-j} = (u_1, \ldots, u_{j-1}, u_{j+1}, \ldots, u_d)^T$ in $[0,1]^{d-1}$ and $u^k_{i_k-1} < u_k < u^k_{i_k}$. Here, $j \in \{1, \ldots, d\}$, $p_{i_j|\mathbf{i}_{-j}}$, and p_{+i_j+} as in Definition 2.6 and Definition 2.5, respectively.

As mentioned in Wei and Kim (2021), we can define the checkerboard copula regression function as follows.

Definition 3.4 (Checkerboard Copula Regression (CCR))**.** (Wei and Kim 2021) The **checkerboard copula regression function** of U_j on $\mathbf{U}_{-j}$ is defined as

$$r_{U_j|\mathbf{U}_{-j}}(\mathbf{u}_{-j}) \equiv E_{c^+}(U_j|\mathbf{U}_{-j} = \mathbf{u}_{-j}) = \int_0^1 u_j c^+(u_j|\mathbf{u}_{-j}) du_j = \sum_{i_j=1}^{I_j} p_{i_j|\mathbf{i}_{-j}} s^j_{i_j}$$

In other words, the CCR function is the mean checkerboard score of X_j with respect to the conditional distribution at the category $\mathbf{i}_{-j}$ of $(d-1)$ explanatory variables $\mathbf{X}_{-j}$.

3.4.1 2-D Example (continued...)

(Adapted from Wei and Kim 2021)

After applying the above definitions, we obtain the following tabular representations of conditional p.m.f.s and checkerboard copula regressions.

Table 3.2: Conditional p.m.f of X_2 given X_1

$X_1 \backslash X_2$	x_1^2	x_2^2	x_3^2
x_1^1	0	0	1
x_2^1	0	1	0
x_3^1	1	0	0
x_4^1	0	1	0
x_5^1	0	0	1

Table 3.3: Conditional p.m.f of X_1 given X_2

$X_1 \backslash X_2$	x_1^2	x_2^2	x_3^2
x_1^1	0	0	1/2
x_2^1	0	1/2	0
x_3^1	1	0	0
x_4^1	0	1/2	0
x_5^1	0	0	1/2

Table 3.4: Checkerboard copula regression of U_2 on U_1

u_1	$r_{U_2\|U_1}(u_1)$
$[0, 2/8]$	12/16
$(2/8, 3/8]$	6/16
$(3/8, 5/8]$	2/16
$(5/8, 6/8]$	6/16
$(6/8, 1]$	12/16

Table 3.5: Checkerboard copula regression of U_1 on U_2

u_2	$r_{U_1\|U_2}(u_2)$
$[0, 2/8]$	1/2
$(2/8, 4/8]$	1/2
$(4/8, 1]$	1/2

3.4.2 Point Prediction Using CCR

The CCR and its prediction are designed to explore and identify the potential regression association between an ordinal response variable and a set of categorical predictors of interest. Thus, we can use Definition 3.4 for predicting the category of the response variable for a given combination of categories of explanatory variables while describing the dependence structure between them.

Suppose that X_j is the response variable, and all the remaining variables in the table (denoted by $\mathbf{X}_{-j}$) are to be used as predictors. Recall Definition 2.6, where we denote the (finite and discrete) range of the marginal distribution of X_j to be $D_j = \{u_0^j, \ldots, u_{i_j}^j, \ldots, u_{I_j}^j\}$. Then, $u_0^j = 0$, $u_{I_j}^j = 1$, and $u_{i_j}^j = \sum_{k_j=1}^{i_j} p_{+k_j+}$. As mentioned in Wei and Kim (2021), we can use this to find $\mathbf{u}_{-j}^*$ from

$Ran(\mathbf{X}_{-j}) = \prod_{k=1,k\neq j}^{d} D_k$. Using this along with Definition 3.4 gives us the estimated value of the checkerboard copula regression, $u_j^* = r_{U_j|\mathbf{U}_{-j}}(\mathbf{u}_{-j}^*)$. Now, using this we can obtain i_j^* and $u_{i_j^*}^j$ such that $u_{i_j^*-1}^j < u_j^* < u_{i_j^*}^j$. This finally leads us to the predicted category $x_{i_j^*}^j$ of the response variable X_j.

In order to better understand this, let us walk through the Example at hand.

3.4.3 2-D Example (continued...)

(Adapted from Wei and Kim 2021)

Upon application of the method detailed above, we can predict the category of X_2 for each category of X_1. For instance, given that $X_1 = x_3^1 = x_{i_1=3}^{1*}$, the corresponding $u_3^{1*} = 5/8$, and thus the predicted value of the CCR is $u_2^* = r_{U_2|U_1}(5/8) = 1/8 \in [0, 2/8]$. This implies that $i_2^* = 1$ and $u_{i_2^*=1}^2 = 2/8$ because $u_0^2 = 0 < u_2^* = 1/8 \leq u_1^2 = 2/8$. Hence, the predicted category of X_2 given $X_1 = x_3^1$ is $f_{X_2|X_1}(x_3^1) = x_1^2$.

After applying this method to all combinations of predictors and response, we obtain the following tabular representations of point predictions through CCR.

Table 3.6: Point prediction through CCR of X_2 on X_1

X_1	u_2^*	$f_{X_2\|X_1}$
x_1^1	6/8	x_3^2
x_2^1	3/8	x_2^2
x_3^1	1/8	x_1^2
x_4^1	3/8	x_2^2
x_5^1	6/8	x_3^2

Table 3.7: Point prediction through CCR of X_1 on X_2

X_2	u_1^*	$f_{X_1\|X_2}$
x_1^2	1/2	x_3^1
x_2^2	1/2	x_3^1
x_3^2	1/2	x_3^1

We can clearly see from the above tables how the prediction results reflect the quadratic relationship shown when we first established this 2-D Example at the start of this chapter.

3.4.4 Empirical Estimation of CCR and Point Prediction

Now, continuing from the notation established in Section 3.3.2, we can estimate CCR for $k \in 1, \ldots, j-1, j+1, \ldots, d$,

$$\hat{r}_{U_j|\mathbf{U}_{-j}}(\mathbf{u}_{-j}) = \sum_{i_j=1}^{I_j} \hat{p}_{i_j|\mathbf{i}_{-j}} \hat{s}_{i_j}^j \text{ for } \hat{u}_{i_k-1} < u_k \leq \hat{u}_{i_k}^k$$

Now, using the above alongside the steps we mentioned in the previous subsection, we can obtain the predicted category of a response variable for each combination of categories of predictors. That is, for a given combination categories of the $(d-1)$-predictors $\mathbf{X}_{-j}$, we find the corresponding $\hat{\mathbf{u}}_{-j}^*$ from the estimated ranges of $\mathbf{X}_{-j}$ and then obtain the estimated value of the CCR as mentioned above, $\hat{u}_j^* = \hat{r}_{U_j|\mathbf{U}_{-j}}(\hat{\mathbf{u}}_{-j}^*)$. From the estimated range of a response variable X_j, we get i_j^* and $\hat{u}_{i_j^*}^j$ such that $\hat{u}_{i_j^*-1}^j < \hat{u}_j^* \leq \hat{u}_{i_j^*}^j$. This implies that the predicted category of X_j is $\hat{x}_{i_j^*}^j$.

More details on these estimators, including asymptotic analysis, are outside the scope of this exposition but are discussed at length in Wei and Kim (2021).

3.4.5 Uncertainty Evaluation of the CCR prediction using nonparametric bootstrap

We can use a nonparametric bootstrap to quantify the uncertainty of the predicted category obtained from CCR. This involves generating multiple bootstrap samples from the original contingency table, computing the checkerboard copula regression for each resampled dataset, and then predicting the category of the response variable for each combination of categories of the explanatory variables.

For each bootstrap sample, we follow these steps:

1. Resample with replacement from the original data to create a bootstrap sample with the same size as the original data.
2. Estimate the checkerboard copula regression based on this bootstrap sample
3. Predict the category of the response variable using the estimated regression
4. Repeat steps 1-3 multiple times (e.g., 1000 times)

The distribution of predicted categories across bootstrap samples measures the prediction uncertainty. We can calculate the proportion of times each category is predicted for a given combination of explanatory variables, with higher proportions indicating greater confidence in the prediction.

For Example, in our 2-D contingency table, using 1000 bootstrap resamples, we quantified the uncertainty of the predicted category of X_2 for each category of X_1. The results showed that the proportion of bootstrap samples where the predicted category matched our original prediction was 100%, indicating high confidence in our predictions.

3.5 Checkerboard Copula Regression Association Measure

Using the CCR discussed above, Wei and Kim (2021) proposed the checkerboard copula regression-based association measure (CCRAM) for a multi-way contingency table with an ordinal response

variable and categorical (ordinal or nominal) explanatory variables.

Definition 3.5 (Checkerboard Copula Regression-Based Association Measure (CCRAM))**.** (Wei and Kim 2021) The **checkerboard copula regression-based association measure (CCRAM)** of X_j on $\mathbf{X}_{-j} = (X_1, \ldots, X_{j-1}, X_{j+1}, \ldots, X_d)^\top$ is

$$\rho^2_{(\mathbf{X}_{-j} \to X_j)} \equiv \frac{\mathrm{Var}[r_{U_j|\mathbf{U}_{-j}}(\mathbf{U}_{-j})]}{\mathrm{Var}(U_j)} = \frac{\mathrm{E}\left[\left(r_{U_j|\mathbf{U}_{-j}}(\mathbf{U}_{-j}) - 1/2\right)^2\right]}{1/12} = 12 \sum_{\mathbf{i}_{-j}=\mathbf{1}}^{\mathbf{I}_{-j}} \left(\sum_{i_j=1}^{I_j} p_{i_j|\mathbf{i}_{-j}} s^j_{i_j} - 1/2 \right)^2 p_{i_1,\ldots,+_j,\ldots,i_d}$$

where $j \in \{1, \ldots, d\}$, and U_j and $\mathbf{U}_{-j} = (U_1, \ldots, U_{j-1}, U_{j+1}, \ldots, U_d)^\top$ are the random variables on $[0,1]^d$ associated with the checkerboard copula density $c^*(\mathbf{u})$ in Definition 2.7.

Extending what is proven for this new measure, Wei and Kim (2021) provides a proposition with proof containing several properties of CCRAM, concluding that:

- CCRAM can identify linear and non-linear relationships between a response variable and several explanatory variables. CCRAM can also be applied when any predictors are nominal and/or a binary response variable.
- CCRAM is lower bounded by zero and upper bounded by $12\sigma^2_{S_j}$, where 0 means no contribution of predictors to the construction of the checkerboard copula regression function.

In order to provide a normalized measure that is independent of the marginal distribution of X_j, Wei and Kim (2021) proposes SCCRAM, which is a scaled version of CCRAM, and it is mathematically defined as follows:

Definition 3.6 (Scaled Checkerboard Copula Regression-Based Association Measure (SCCRAM))**.** (Wei and Kim 2021) The **scaled checkerboard copula regression-based association measure**

(SCCRAM) of X_j on $\mathbf{X}_{-j} = (X_1, \ldots, X_{j-1}, X_{j+1}, \ldots, X_d)^\top$ is defined as

$$\rho^{2*}_{(\mathbf{X}_{-j} \to X_j)} = \frac{\rho^2_{(\mathbf{X}_{-j} \to X_j)}}{12\sigma^2_{S_j}}$$

where $\rho^2_{(\mathbf{X}_{-j} \to X_j)}$ and $\sigma^2_{S_j}$ are defined as in Definition 3.5 and Lemma 3.1 respectively.

(S)CCRAM is designed to quantify the regression association identified by Checkerboard Copula Regression and its prediction.

3.5.1 2-D Example (continued...)

(Adapted from Wei and Kim 2021)

Using the above definitions, we obtain: $(\rho^2_{(X_1 \to X_2)}, 12\sigma^2_{S_2}, \rho^{2*}_{(X_1 \to X_2)}) = (27/32, 27/32, 1)$ and $(\rho^2_{(X_2 \to X_1)}, 12\sigma^2_{S_1}, \rho^{2*}_{(X_2 \to X_1)}) = (0, 243/256, 0)$.

$\rho^{2*}_{(X_1 \to X_2)} = 1$ implies that X_1 perfectly explains the variation in X_2 induced by its checkerboard copula score and its marginal distribution and this result agrees with the observation that $r_{U_2|U_1}(u_1)$ equals one and only one of the checkerboard score of X_2. This result also supports that X_2 functionally depends on X_1 with probability 1. On the other hand, $\rho^{2*}_{(X_2 \to X_1)} = 0$ means that $r_{U_1|U_2}(u_2) = E(U_1) = 1/2, \forall u_2$. Thus, X_2 has no contribution arising from its score and marginal distribution in explaining the variation in X_1.

3.5.2 Empirical Estimation of (S)CCRAM

Now, continuing from the notation established in Section 3.3.2, the estimators for the CCRAM and SCCRAM, as defined in Definition 3.5 and Definition 3.6, respectively, are given below:

$$\hat{\rho}^2_{(\mathbf{X}_{-j}\to X_j)} = 12 \sum_{\mathbf{i}_{-j}=\mathbf{1}}^{\mathbf{I}_{-j}} \left(\sum_{i_j=1}^{I_j} \hat{p}_{i_j|\mathbf{i}_{-j}} \hat{s}^d_{i_j} - \frac{1}{2} \right)^2 \hat{p}_{i_1,\ldots,+_j,\ldots,i_d}, \qquad \hat{\rho}^{2*}_{(\mathbf{X}_{-j}\to X_j)} = \frac{\hat{\rho}^2_{(\mathbf{X}_{-j}\to X_j)}}{12\hat{\sigma}^2_{S_j}}$$

3.5.3 Uncertainty Evaluation of the estimated (S)CCRAM using nonparametric bootstrap distribution and its confidence interval

We can employ a nonparametric bootstrap to generate an empirical sampling distribution to assess the uncertainty of the estimated (S)CCRAM. This approach involves:

1. Generate B bootstrap samples of the same size as the original data (typically $B = 1000$) by resampling with replacement from the original data.

2. For each bootstrap sample $b = 1, \ldots, B$, compute the estimated (S)CCRAM, denoted as $\hat{\rho}^2_{(\mathbf{X}_{-j}\to X_j),b}$ and $\hat{\rho}^{2*}_{(\mathbf{X}_{-j}\to X_j),b}$.

3. Construct the empirical bootstrap distribution of the estimates $\{\hat{\rho}^2_b\}_{b=1}^B$.

From this bootstrap distribution, we can calculate the bootstrap standard error as the standard deviation of the bootstrap estimates and construct confidence intervals using various methods:

- Percentile Method (Davison and Hinkley 1997): The $(1-\alpha)\times 100\%$ confidence interval is given by the $\alpha/2$ and $1-\alpha/2$ quantiles of the bootstrap distribution: $\left[\hat{\rho}^2_{\alpha/2}, \hat{\rho}^2_{1-\alpha/2}\right]$

- Basic (Reverse Percentile) Method (Hesterberg 2014): This method reflects the bootstrap quantiles around the original estimate to produce the interval: $\left[2\hat{\rho}^2 - \hat{\rho}^2_{1-\alpha/2},\ 2\hat{\rho}^2 - \hat{\rho}^2_{\alpha/2}\right]$, where $\hat{\rho}^2$ is the (S)CCRAM estimate from the original dataset.

- BCa (Bias-Corrected and Accelerated) Method (Efron 1987): This method adjusts the interval to correct for both bias and skewness in the bootstrap distribution, using acceleration and bias-correction terms derived from the data. It offers better coverage properties, especially in

small samples or skewed distributions.

These confidence intervals provide a principled way to quantify the precision of the (S)CCRAM estimates, offering insight into the variability due to sampling and helping assess the statistical significance of regression dependence.

3.5.4 Statistical significance of the estimated (S)CCRAM using Permutation distribution and its hypothesis testing

In the case of CCRAM, Wei and Kim (2021) proposes permutation testing for null hypothesis $H_0 : \rho^2_{(\mathbf{X}_{-j} \to X_j)} = 0$, which indicates no association between the response variable and the explanatory variables.

The permutation testing procedure involves the following:

1. Calculate the observed (S)CCRAM value for the original data
2. Generate M permutation samples (typically $M = 10^6$) by randomly permuting the response variable values while keeping the explanatory variables fixed, thus breaking any potential association
3. For each permutation sample $m = 1, \ldots, M$, compute the (S)CCRAM, denoted as $\hat{\rho}^2_{(\mathbf{X}_{-j} \to X_j),m}$ and $\hat{\rho}^{2*}_{(\mathbf{X}_{-j} \to X_j),m}$
4. Construct the empirical permutation distribution of the estimates under the null hypothesis
5. Calculate the p-value as the proportion of permutation statistics that are as extreme as or more extreme than the observed statistic:

$$p\text{-value} = \frac{1}{M} \sum_{m=1}^{M} I(\hat{\rho}^2_{(\mathbf{X}_{-j} \to X_j),m} \geq \hat{\rho}^2_{(\mathbf{X}_{-j} \to X_j),obs})$$

Suppose the p-value is less than a predetermined significance level (e.g., 0.05). In that case, we reject the null hypothesis and conclude that there is a significant association between the response variable and the explanatory variables.

This permutation approach provides a distribution-free method for hypothesis testing, analogous to testing $R^2 = 0$ in linear regression, but appropriate for the categorical data context of the checkerboard copula regression.

3.6 Visualization Methods

Effective visualization is crucial for understanding and interpreting the regression associations identified by checkerboard copula regression. Liao et al. (2024) details several visualization approaches particularly suited for displaying the dependence structures in multi-dimensional contingency tables with an ordinal response variable.

3.6.1 Cross-tabulation

For simpler two-way contingency tables (as seen in the ice cream study example within Liao et al. (2024)), cross-tabulation provides a straightforward approach to visualize the predicted categories of the response variable for each category of the explanatory variable. One can enhance these tables by color-coding the predicted categories and including bootstrap proportions to represent prediction uncertainty. This allows for a clear comparison between the observed regression pattern and the pattern expected under independence.

3.6.2 Bubble Plots

When dealing with higher-dimensional contingency tables (such as the back pain data with three explanatory variables in Wei and Kim (2021)), bubble plots offer a practical visualization approach.

In these plots, the x-axis represents different combinations of categories of explanatory variables, while the y-axis shows the categories of the response variable. Dark dots indicate the predicted category for each combination, and bubbles (circles) with varying sizes represent the proportion of times each category is predicted across bootstrap samples. This visualization clearly reveals complex association patterns, such as potential interaction effects among explanatory variables.

3.6.3 Doubledecker Plots

Doubledecker plots provide a particularly insightful visualization for multi-dimensional contingency tables with temporal or hierarchical structure (as in Three Mile Island data within Liao et al. (2024)). These plots display vertical splits for explanatory variables and horizontal splits for the response variable. The width of each bar is proportional to the observed frequency of the corresponding combination of explanatory variables. In contrast, the heights of color-coded blocks within each bar represent the proportions of predicted categories across bootstrap samples. This approach effectively visualizes both the magnitude and uncertainty of predictions while accounting for the natural ordering or hierarchy among variables.

All these visualization methods can be paired with corresponding "null reference" plots generated through permutation methods, allowing researchers to visually assess whether the detected regression patterns significantly differ from those expected under independence. This combination of exploratory visualization and resampling-based calibration provides a comprehensive framework for understanding regression dependence in categorical data without relying on parametric assumptions.

Chapter 4

Software (Package) Implementation and Testing

In this chapter, we introduce `ccrvam`, a Python package that implements the Checkerboard Copula Regression-based Visualization and Association Measure (CCRVAM) techniques discussed in previous chapters. Despite the growing importance of multivariate categorical data analysis with ordinal response variables in disciplines like medicine, social sciences, and economics, there has been a notable lack of user-friendly, well-tested software implementations that scale efficiently to higher-dimensional problems. The `ccrvam` package addresses this gap by providing a comprehensive suite of tools for analyzing multivariate discrete data using the checkerboard copula approach.

4.1 Set-up and Example Data

The `ccrvam` package is built to analyze multi-dimensional contingency tables with an ordinal response variable and a set of categorical (nominal/ordinal) explanatory variables/predictors. This aligns perfectly with the theoretical frameworks established in Chapter 3. The package is designed with ease of installation and use in mind, particularly within Jupyter Notebook environments common in data analysis workflows.

4.1.1 Installation

For quick use in Jupyter Notebooks, the package can be installed directly from PyPI with:

```
pip install ccrvam==1.0.0
```

This thesis uses and demonstrates functionality based on version `v1.0.0` of the `ccrvam` package. While this version captures the stable features implemented during the course of this work, the package remains under active development. The most recent release, along with version history and updates, is available on pypi.org/project/ccrvam.

For more containerized production-heavy work, a custom virtual environment is recommended. Instructions for setting that up can be found at ccrvam/.github/README.md.

4.2 Types of Input Data Supported

The `ccrvam` package is designed to be flexible with respect to input data formats. The package supports these as the main formats:

4.2.1 Loading from Contingency Table Format (In-Place)

The `ccrvam` package implements the theoretical framework outlined in Chapter 3 by operating on multi-dimensional contingency tables. These tables represent the joint distribution of categorical variables where one variable is explicitly designated as an ordinal response (dependent variable), one or more variables function as categorical predictors (independent variables), and table entries contain frequency counts of observations. The package accepts these contingency tables directly as NumPy arrays, with each dimension corresponding to the categories of a particular variable. This structure allows for efficient computation of the checkerboard copula scores and subsequent

association measures while preserving the natural ordering of the response variable categories and accommodating multiple categorical predictors simultaneously.

For example, consider the 2-D example from the previous chapter:

```
import numpy as np
from ccrvam import GenericCCRVAM

# Migraine treatment example (dose vs. pain severity)
contingency_table_2d = np.array([
    [0, 0, 20],
    [0, 10, 0],
    [20, 0, 0],
    [0, 10, 0],
    [0, 0, 20]
])

# Create a CCRVAM object
ccrvam_obj = GenericCCRVAM.from_contingency_table(contingency_table_2d)
# Dimension of the inferred joint probability matrix P:
print(ccrvam_obj.P.shape)
```

```
(5, 3)
```

```
# Joint Probability matrix P:
print(ccrvam_obj.P)
```

```
[[0.    0.    0.25 ]
 [0.    0.125 0.   ]
 [0.25  0.    0.   ]
 [0.    0.125 0.   ]
 [0.    0.    0.25 ]]
```

For higher dimensions, the package supports multi-dimensional NumPy arrays representing contingency tables across multiple predictors. More examples in this regard can be found in Chapter 5.

4.2.2 Loading from External Data Files

Through the `DataProcessor` class in our package (`ccrvam`), we support the flexible loading of categorical data in multiple formats, accommodating diverse data structures commonly encountered in statistical analysis. The class provides a unified interface for importing data regardless of its original format, making it accessible for CCRVAM analysis without requiring extensive preprocessing.

The `DataProcessor` class supports three primary data formats:

1. **Case-form data:** Where each row represents an individual case with categorical variables organized in separate columns, allowing for straightforward representation of raw survey or experimental results.

2. **Frequency-form data:** Each row contains a unique combination of categorical variables and their corresponding frequency count, offering a more compact representation when many observations share identical category combinations.

3. **Table-form data:** Where direct contingency tables are represented as multi-dimensional arrays, providing the most computationally efficient input format when other statistical software aggregates data into contingency tables.

The implementation also includes robust handling of several key data management features. The package supports custom variable naming schemes, allowing users to define meaningful labels for their variables rather than relying on default numeric identifiers. It automatically maps non-integer category values to integer indices, enabling seamless processing of categorical data with text or mixed-type labels. The software accommodates custom delimiters for text-based input files, offering flexibility when importing data from various sources with different formatting conventions. Additionally, the implementation supports the dimensional specification for proper array structuring,

ensuring that contingency tables are correctly shaped according to the number of categories in each variable. This is essential for accurate computation of the checkerboard copula scores and associated.

This flexible approach to data loading ensures compatibility with various data collection methodologies and storage formats, allowing researchers to focus on analysis rather than data conversion. The `DataProcessor` integrates seamlessly with the `GenericCCRVAM` class to initialize model objects directly from imported data, creating a streamlined workflow from raw data to statistical inference. Detailed examples demonstrating each data format, corresponding code snippets, and implementation considerations are provided in Chapter 5.

4.3 Checkerboard copula score (for ordinal response variable)

Following the theoretical foundation in Definition 3.1, the package implements checkerboard copula scores for ordinal variables. These scores represent a transformation that leverages the inherent ordering information in ordinal variables.

The implementation maintains fidelity to the mathematical definitions while providing a computationally efficient vectorized implementation:

```
# Calculate and display CCS for X1
scores_X1 = ccrvam_obj.calculate_ccs(1)
print(scores_X1)
```

```
[0.125, 0.3125, 0.5, 0.6875, 0.875]
```

```
# Calculate and display the Variance of CCS for X1
variance_ccs_X1 = ccrvam_obj.calculate_variance_ccs(1)
print(variance_ccs_X1)
```

```
0.0791015625
```

Recall that for each ordinal variable X_j with categories $i_j \in \{1, \ldots, I_j\}$, the scores $s^j_{i_j} = (u^j_{i_j-1} + u^j_{i_j})/2$ are calculated where $u^j_{i_j}$ is defined by the marginal cumulative distribution. This implementation follows directly from the empirical estimation procedure detailed in Section 3.3.2. We can see how the output from the above code-chunk matches the result in our running example from Chapter 3, which further verifies the reproducible functionality of our package.

`ccrvam` employs vectorized operations through NumPy (Developers 2025) to ensure computational efficiency, which becomes particularly important for higher-dimensional tables. The variance calculation implements Lemma 3.1, providing a measure of dispersion that is essential for the scaled association measures discussed later.

4.4 Checkerboard copula Regression (CCR)

The Checkerboard Copula Regression functionality follows the definition provided in Definition 3.4, computing the conditional expectation of the copula score for the response variable given values of the predictor variables.

The prediction functionality follows the empirical estimation procedure described in #sec-empirical-ccr, where the predicted category $\hat{x}^j_{i^*_j}$ is determined by finding the interval containing the estimated regression value $\hat{u}^*_j = \hat{r}_{U_j|\mathbf{U}_{-j}}(\hat{\mathbf{u}}^*_{-j})$.

The `get_category_predictions_ccr()` method performs the essential function of predicting the categories of the response variable (specified through the response input argument) based on given predictor values (enumerated in the `predictors` input argument). This method implements the core predictive capability of the checkerboard copula regression approach, translating theoretical associations into practical category predictions. The method returns these predictions in an easy-to-read Pandas (Mckinney 2011) DataFrame format, making it straightforward for researchers to

examine and interpret the results in a familiar tabular structure. Additionally, the method supports custom variable names for enhanced interpretation. It allows users to replace default numeric identifiers with meaningful labels that reflect the actual variables being analyzed in their specific domain context.

The implementation also allows for multiple conditioning axes, supporting complex multivariate analyses, as seen in the examples mentioned in Section 4.12.2.

```
# Predictions from X1 to X2:
predictions_X1_to_X2 = ccrvam_obj.get_predictions_ccr(
    predictors=[1],
    response=2
)
print(predictions_X1_to_X2)
```

	X1 Category	Predicted Response Category
0	1	3
1	2	2
2	3	1
3	4	2
4	5	3

```
# Example: Showcasing the use of custom variable names for the output
# Predictions from Education Level to Income Bracket:
variable_to_name_dict = {
    1: "Income",
    2: "Education"
}
predictions_Education_to_Income = ccrvam_obj.get_predictions_ccr(
    predictors=[2],
    response=1,
    variable_names=variable_to_name_dict
)
print(predictions_Education_to_Income)
```

	Education Category	Predicted Income Category

```
0                    1                    3

1                    2                    3

2                    3                    3
```

The package also provides reference prediction under joint independence, which is important for interpreting the substantive meaning of predictions by comparing them against what would be expected if no association existed.

Hence, we can also obtain the response category prediction under the assumption of joint independence between X_1 and X_2 as follows:

```python
# Response category prediction under the joint independence between X1 and X2
print(ccrvam_obj.get_prediction_under_indep(2))
```

```
2
```

4.5 CCR Predicted Category Visualization

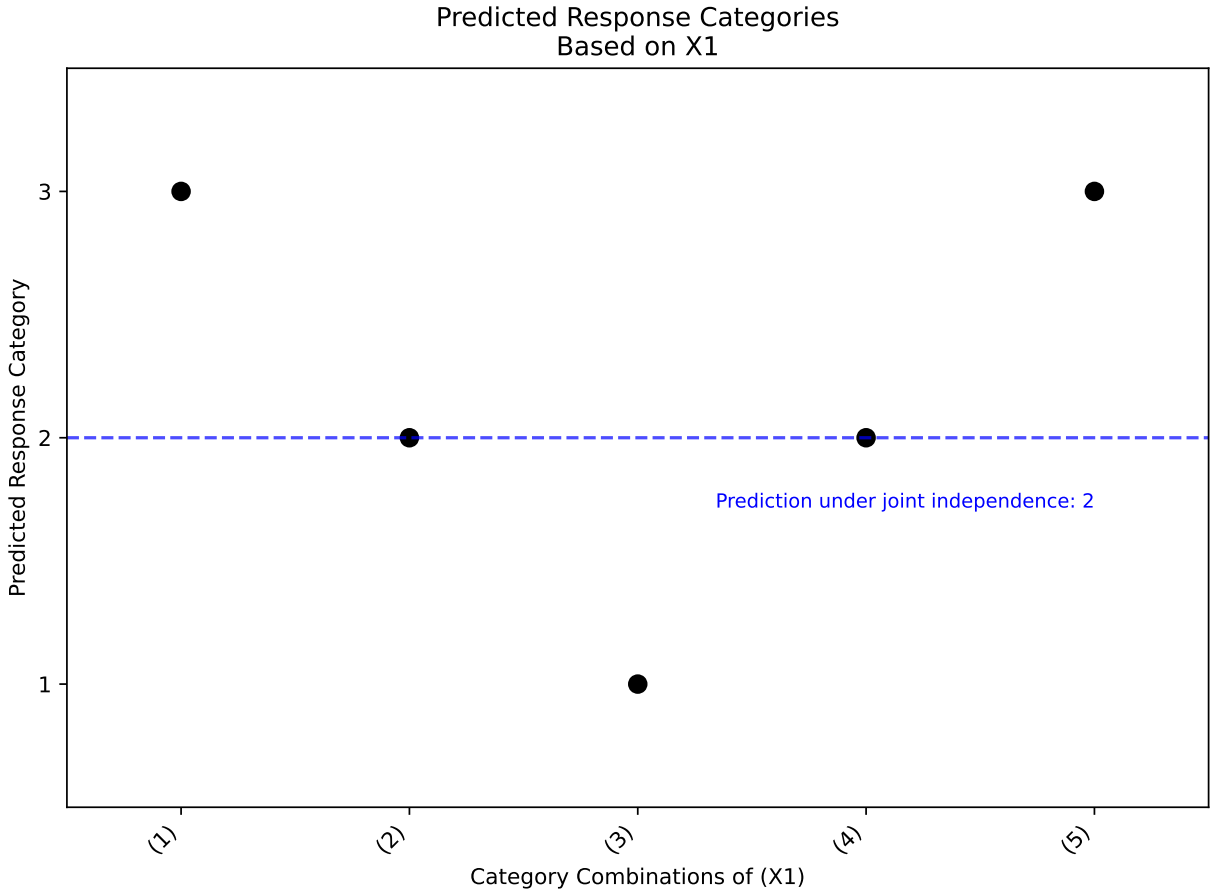
The `ccrvam` package includes a comprehensive set of visualization tools for exploring dependence structures in multivariate ordinal data. The package provides a built-in visualization method for CCR predictions:

This visualization approach creates a heatmap-style plot showing the predicted categories of the response variable for different combinations of predictor variable categories. It includes markers for predicted categories and optional reference lines for predictions under joint independence.

The visualization methods support various color schemes for different visual preferences, customizable figure sizes and resolutions, text annotations showing prediction values, different legend styles for handling many predictor combinations, and exportable high-resolution graphics for publications.

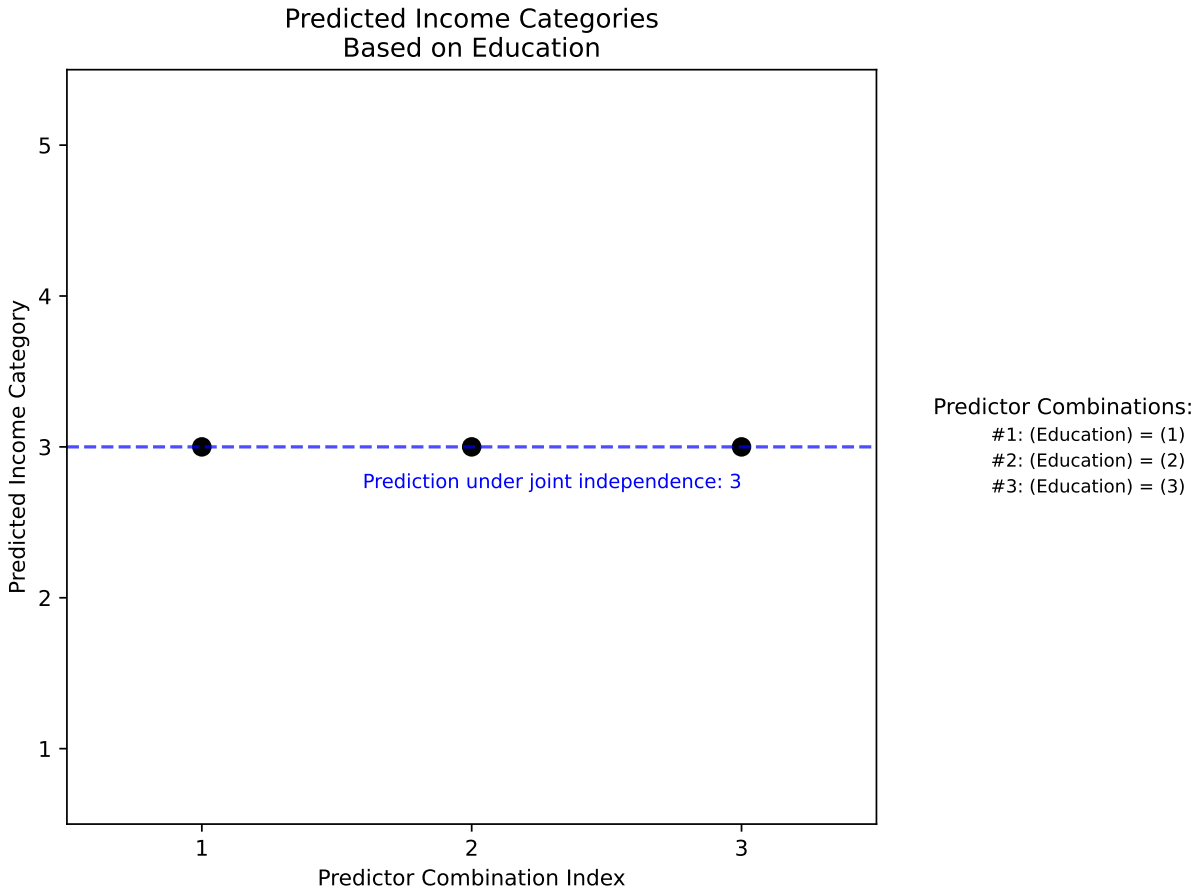
These visualizations help researchers understand and communicate the complex dependence structures detected by the CCR approach, making the results more accessible and interpretable.

```
# Plotting with default naming scheme with tuple labels on the x-axis
ccrvam_obj.plot_ccr_predictions(
  predictors=[1],
  response=2,
  legend_style="xaxis"
)
```




```
# Plotting with the custom naming scheme with the legend of category combinations
var_names={1: "Income", 2: "Education"}

ccrvam_obj.plot_ccr_predictions(
  predictors=[2],
  response=1,
  legend_style="side",
  variable_names=var_names
)
```



4.6 CCR Prediction Uncertainty Evaluation Using Nonparametric Bootstrap Resampling

To quantify prediction uncertainty, the package implements nonparametric bootstrap methods:

```
from ccrvam import bootstrap_predict_ccr_summary

prediction_matrix = bootstrap_predict_ccr_summary(
    contingency_table_2d,
    predictors=[1],
    predictors_names=["X"],
    response=2,
    response_name="Y",
    n_resamples=9999
)
```

```
# Predictions Summary Matrix
print(prediction_matrix)
```

```
       Y=1    Y=2    Y=3
X=1    0.0    0.0  100.0
X=2    0.0  100.0    0.0
X=3  100.0    0.0    0.0
X=4    0.0  100.0    0.0
X=5    0.0    0.0  100.0
```

```
print(prediction_matrix.predictions)
```

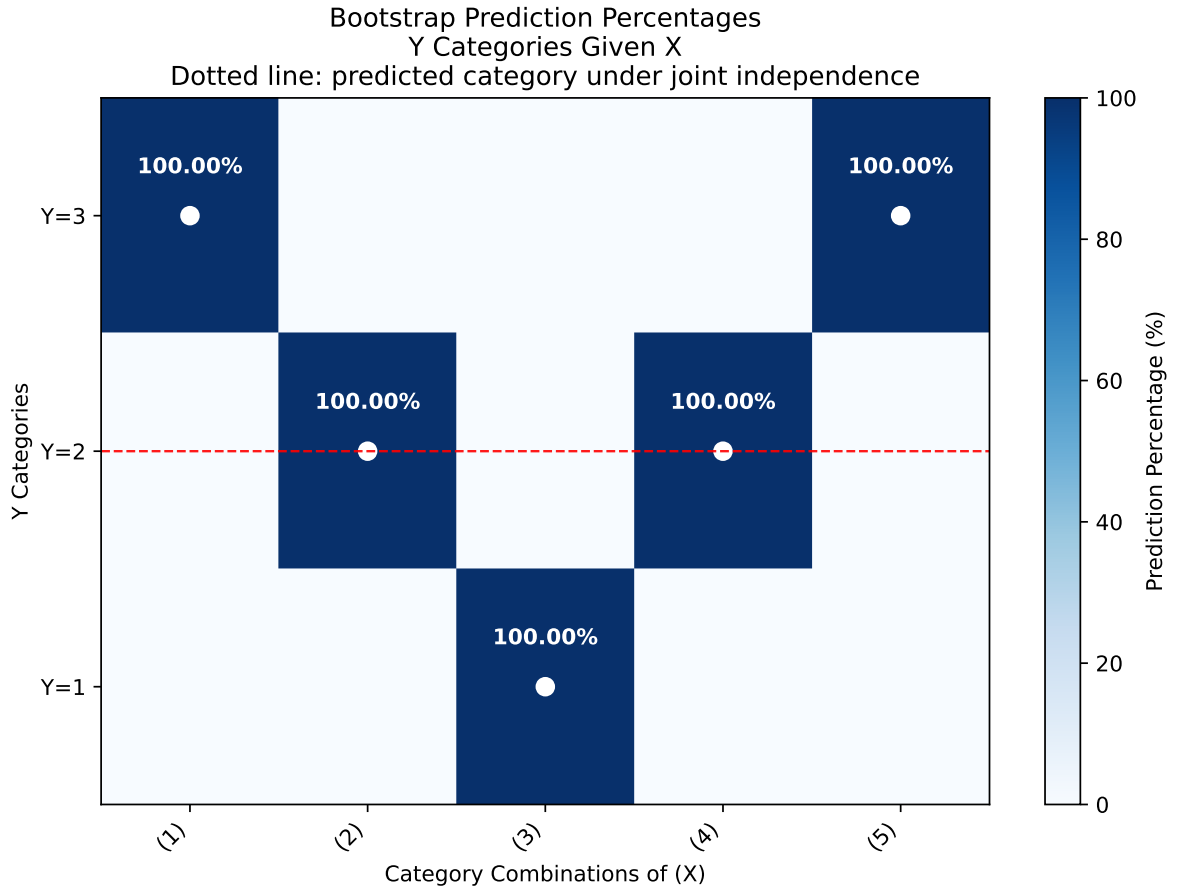
```
    Predicted
X=1         3
X=2         2
X=3         1
X=4         2
X=5         3
```

This implementation follows the nonparametric bootstrap procedure outlined in the previous chapter, where multiple bootstrap samples are generated from the original contingency table, and predictions are made for each resampled dataset. The distribution of predicted categories provides a measure of prediction uncertainty, represented as percentages in the resulting heatmap visualization.

The visualization component employs a color gradient to represent the confidence in predictions, with darker colors indicating higher prediction percentages (greater confidence). Dotted lines indicate predictions under joint independence, providing a reference point for interpretation. More input arguments and options for customization can be explored further in ccrvam.readthedocs.io, which

hosts detailed documentation for our `ccrvam` package.

```
# We can also visualize the results with the attached plotting method
prediction_matrix.plot_prediction_heatmap()
```



4.7 (S)CCRAM Estimation

The package implements both the unscaled (CCRAM) and scaled (SCCRAM) versions of the checkerboard copula regression association measure, as defined in Definition 3.5 and Definition 3.6:

```
ccram_X1_to_X2 = ccrvam_obj.calculate_CCRAM(
                    predictors=[1],
                    response=2
                )
print(f"CCRAM X1 to X2: {ccram_X1_to_X2:.4f}")
```

```
CCRAM X1 to X2: 0.8438
```

```
sccram_X1_to_X2 = ccrvam_obj.calculate_CCRAM(
                    predictors=[1],
                    response=2,
                    scaled=True
                  )
print(f"SCCRAM X1 to X2: {sccram_X1_to_X2:.4f}")
```

```
SCCRAM X1 to X2: 1.0000
```

The implementation follows the empirical estimation procedures outlined in the previous chapter. CCRAM measures the proportion of variance in the response variable's checkerboard copula score that the predictor variables can explain. SCCRAM normalizes this measure to be bounded between 0 and 1, making it easier to interpret and compare across different datasets.

Both measures quantify the strength of even the nonlinear regression relationship between the ordinal response variable and categorical predictors, going beyond traditional correlation measures that primarily detect linear relationships.

4.8 (S)CCRAM Uncertainty Evaluation Using Bootstrap Resampling

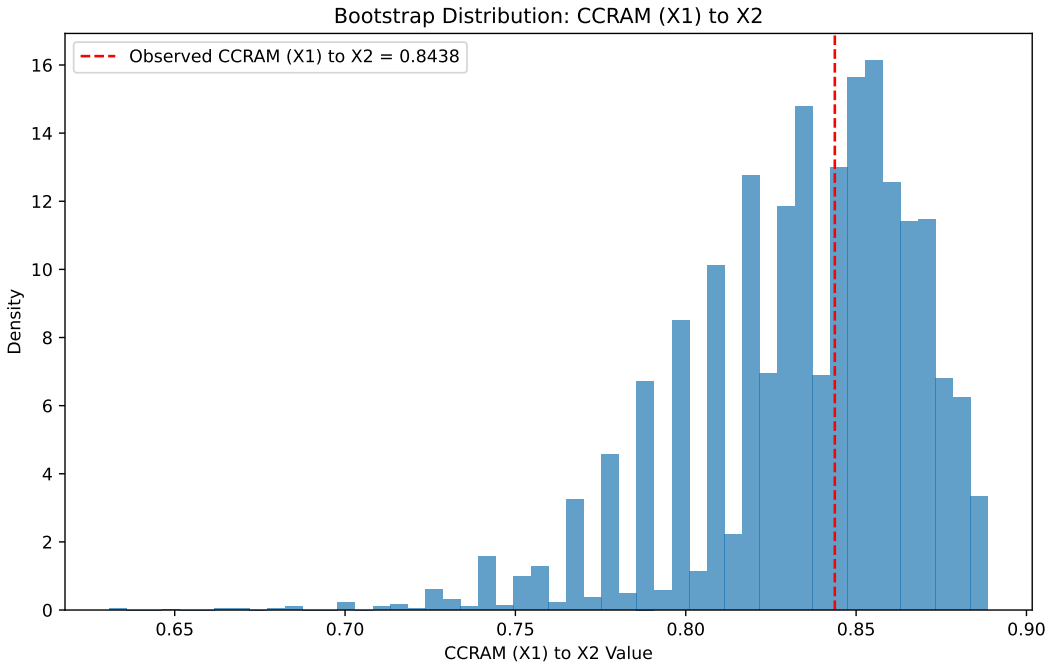
As mentioned in Section 3.5.3, in order to assess the precision of CCRAM and SCCRAM estimates, the package implements nonparametric bootstrap procedures. `ccrvam` incorporates nonparametric bootstrap procedures based on the `scipy.stats.bootstrap` function (Virtanen et al. 2020). This approach allows users to generate confidence intervals and visualize the empirical distribution of estimated dependence measures, improving inference in discrete multivariate settings.

The bootstrap procedure follows three steps: (1) generating multiple resampled datasets from the original contingency table with replacement; (2) computing the (S)CCRAM statistic for each resample using a custom `ccram_stat` function; and (3) calculating confidence intervals based on the resulting

distribution, following the methodology of user's choice ("percentile", "BCa", "basic") from SciPy (Virtanen et al. 2020).

```
from ccrvam import bootstrap_ccram

ccram_result = bootstrap_ccram(
    contingency_table_2d,
    predictors=[1],
    response=2,
    n_resamples=9999,
    scaled=False,
    confidence_level=0.95,
    method="percentile",
    random_state=None
)
```




```
# Metric Name
print(ccram_result.metric_name)
```

```
CCRAM (X1) to X2
```

```python
# Observed Value
print(f"{ccram_result.observed_value:.4f}")
```

```
0.8438
```

```python
# 95% Confidence Interval
lower_CI_bound = ccram_result.confidence_interval[0]
upper_CI_bound = ccram_result.confidence_interval[1]
print(f"({lower_CI_bound:.4f}, {upper_CI_bound:.4f})")
```

```
(0.7553, 0.8817)
```

```python
# Standard Error
print(f"{ccram_result.standard_error:.4f}")
```

```
0.0328
```

```python
# Bootstrap Estimates
bootstrap_estimates = ccram_result.bootstrap_distribution
print(f"{type(bootstrap_estimates)}")
```

```
<class 'numpy.ndarray'>
```

```python
# Calculate bootstrap bias
bootstrap_mean = np.mean(bootstrap_estimates)
bootstrap_bias = bootstrap_mean - ccram_result.observed_value

# Calculate the bootstrap standard error
bootstrap_std_error = np.std(bootstrap_estimates, ddof=1)

# Calculate the ratio of bias to standard error
bias_to_se_ratio = bootstrap_bias / bootstrap_std_error

# Additional Bootstrap Statistics
print(f"Bootstrap Mean: {bootstrap_mean:.4f}")
```

```
Bootstrap Mean: 0.8356
```

```python
print(f"Bootstrap Bias: {bootstrap_bias:.4f}")
```

```
Bootstrap Bias: -0.0082
```

```
print(f"Bootstrap Standard Error: {bootstrap_std_error:.4f}")
```

```
Bootstrap Standard Error: 0.0328
```

```
print(f"Bias to Standard Error Ratio: {bias_to_se_ratio:.4f}")
```

```
Bias to Standard Error Ratio: -0.2502
```

The bootstrap procedure generates multiple resamples from the original contingency table, calculates each resample's (S)CCRAM, and constructs confidence intervals based on the resulting distribution. This provides a measure of the sampling variability and precision of the (S)CCRAM estimate. The implementation builds directly on SciPy's `bootstrap` function, which allows users to specify the statistic of interest as a callable and automatically handles paired resampling, confidence interval construction, and standard error estimation. We set `vectorized=True` during our SciPy API call to enable fast computation over multiple resamples. As for the above example, we use `method="percentile"` to derive confidence intervals from the empirical quantiles of the bootstrap distribution. But user is allowed to use `"basic"` or `"BCa"` depending on their needs and preferences.

The visualization component plots the bootstrap distribution with the observed (S)CCRAM value highlighted, providing a graphical representation of the uncertainty in the estimate. The auto-plotting in the `CustomBootstrapResult` object highlights the observed value within the distribution, offering a visual representation of uncertainty.

This analysis can be repeated for SCCRAM by setting `scaled = True` as an input argument of the `bootstrap_ccram()` function.

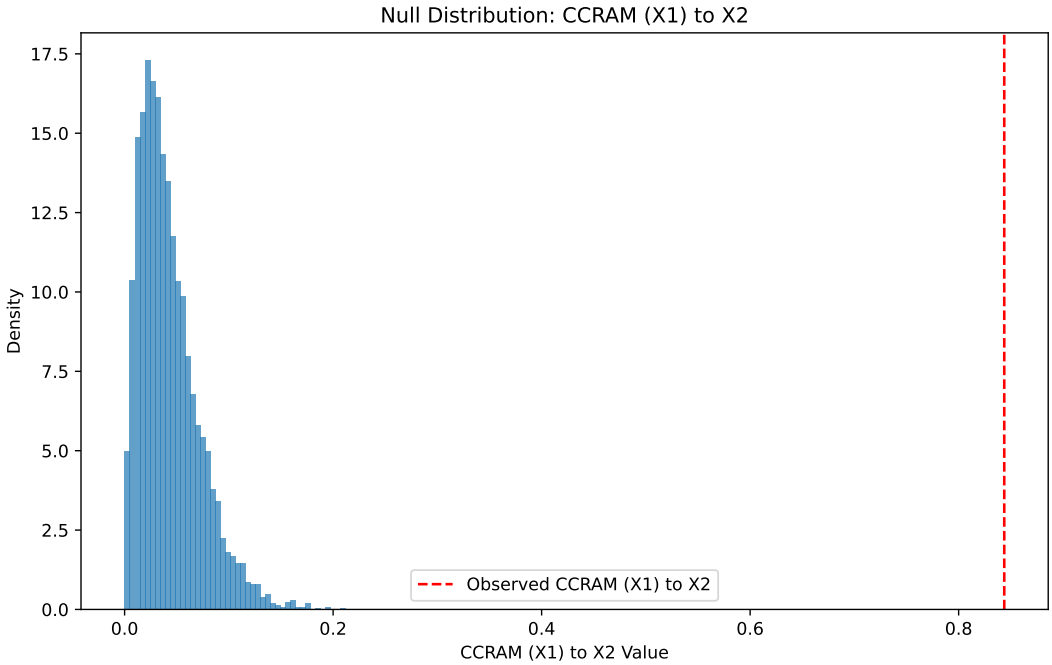
4.9 Statistical Significance Testing for (S)CCRAM Using Permutation Test

As mentioned in Section 3.5.4, the package also implements permutation testing to assess the statistical significance of CCRAM or SCCRAM. We use permutation testing functionality `scipy.stats.permutation_test` from SciPy (Virtanen et al. 2020). This procedure randomly permutes the response labels while keeping the predictor structure fixed, forming a null distribution against which the observed statistic is compared.

We demonstrate the usage of the same within our package `ccrvam` in the case of the 2-D example below:

```
from ccrvam import permutation_test_ccram

perm_result = permutation_test_ccram(
    contingency_table_2d,
    predictors=[1],
    response=2,
    scaled=False,
    alternative='greater',
    n_resamples=9999
)
```




```
print(f"Metric Name: {perm_result.metric_name}")
```

```
Metric Name: CCRAM (X1) to X2
```

```
print(f"Observed Value: {perm_result.observed_value:.4f}")
```

```
Observed Value: 0.8438
```

```
print(f"P-Value: {perm_result.p_value:.4f}")
```

```
P-Value: 0.0001
```

```
# Permutation Distribution
permutation_distribution = perm_result.null_distribution
print(f"Permutation Distribution (Type): {type(permutation_distribution)}")
```

```
Permutation Distribution (Type): <class 'numpy.ndarray'>
```

```python
# Calculate quantiles
q01 = np.quantile(permutation_distribution, 0.01)
# 0.5-th quantile (median)
median = np.median(permutation_distribution)
# 0.99-th quantile
q99 = np.quantile(permutation_distribution, 0.99)

# Calculate interquartile range (IQR)
q25 = np.quantile(permutation_distribution, 0.25)
q75 = np.quantile(permutation_distribution, 0.75)
iqr = q75 - q25

# Permutation Distribution Summary Statistics:
print(f"0.01-th Quantile: {q01:.4f}")
```

```
0.01-th Quantile: 0.0034
```

```python
print(f"0.5-th Quantile (Median): {median:.4f}")
```

```
0.5-th Quantile (Median): 0.0366
```

```python
print(f"0.99-th Quantile: {q99:.4f}")
```

```
0.99-th Quantile: 0.1330
```

```python
print(f"Interquartile Range (IQR): {iqr:.4f}")
```

```
Interquartile Range (IQR): 0.0366
```

This implementation follows the permutation testing procedure outlined in the Chapter 3, where the response variable values are randomly permuted to break any association with the predictor variables, thus generating a null distribution under the hypothesis of no association. The p-value is calculated as the proportion of permutation statistics that is as extreme as or more extreme than the observed statistic. The permutation test is powered by SciPy's `permutation_test` utility (Virtanen et al. 2020). By specifying `permutation_type="pairings"` in our SciPy API call, we ensure that only the response values are shuffled, preserving the dependency structure among predictors. The

test statistic is defined through a wrapper around the CCRAM computation, and the passage of `vectorized=True` in our SciPy API call enables batch execution across resamples.

The visualization component plots the null distribution with the observed CCRAM value highlighted, providing a graphical representation of the statistical significance. As for the above example, we use `alternative="greater"` for our hypothesis testing. But user is allowed to use `"two-sided"` or `"less"` depending on their needs and preferences.

This analysis can be repeated for SCCRAM by setting `scaled = True` as an input argument of the `permutation_test_ccram()` function.

4.10 Software Architecture and Design Principles

The `ccrvam` Python package was developed following modern software engineering principles to ensure reliability, maintainability, and extensibility. We used `PyPi-Template` (VG 2024) to initialize the skeleton of our software package.

4.10.1 Component Structure and Object-Oriented Design

The package follows an object-oriented design, encapsulating related functionality within classes. This design allows users to work with a unified interface while hiding the pesky implementation details from the user, therefore making the package intuitive to use while maintaining flexibility.

The package is organized into three main components:

1. **Core CCRVAM Implementation** (`GenericCCRVAM` class within the `gencopula` module): This central object implements the fundamental calculations for checkerboard copula regression while handling internal data representation and transformation. Through this, we provide our users with several methods for estimating prediction and association measures.

2. **Statistical Simulation Framework** (`genstatsim` module): This module implements bootstrap and permutation testing procedures while providing uncertainty quantification for predictions and measures. Through this, we also provide users flexibility through visualization and exporting methods for statistical results

3. **Data Processing Utilities** (`utils` module): This module handles user-facing data loading and formatting methods, providing conversion between data representations such as table form, case form, and frequency form as outlined in Chapter 5. Through this, we also provide users with basic data validation and preprocessing functionality.

We can visualize the package structure and user-experience-workflow through the images below powered by Mermaid (Sveidqvist and Mermaid 2014):

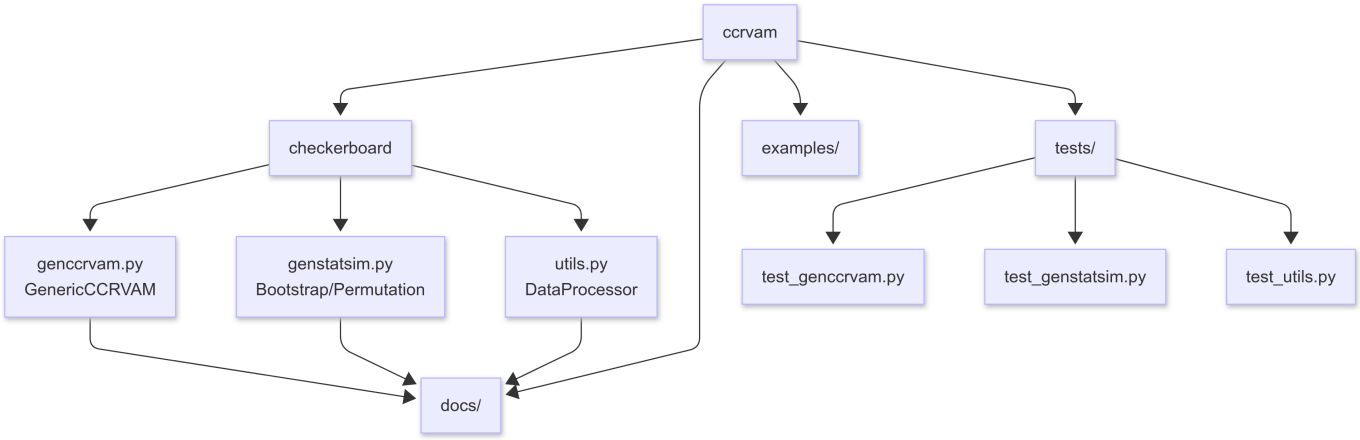


Figure 4.1: High-Level Package Structure of CCRVAM

Figure 4.1 illustrates the modular organization of the `ccrvam` codebase. The core functionality is encapsulated within the checkerboard subpackage, which houses the main analytical engine (`genccrvam.py`), statistical simulation tools (`genstatsim.py`), and data preprocessing utilities (`utils.py`). The top-level package initialization file (`__init__.py`) exposes these modules for external use, while supplementary materials such as documentation, examples, and tests are organized into their respective directories.

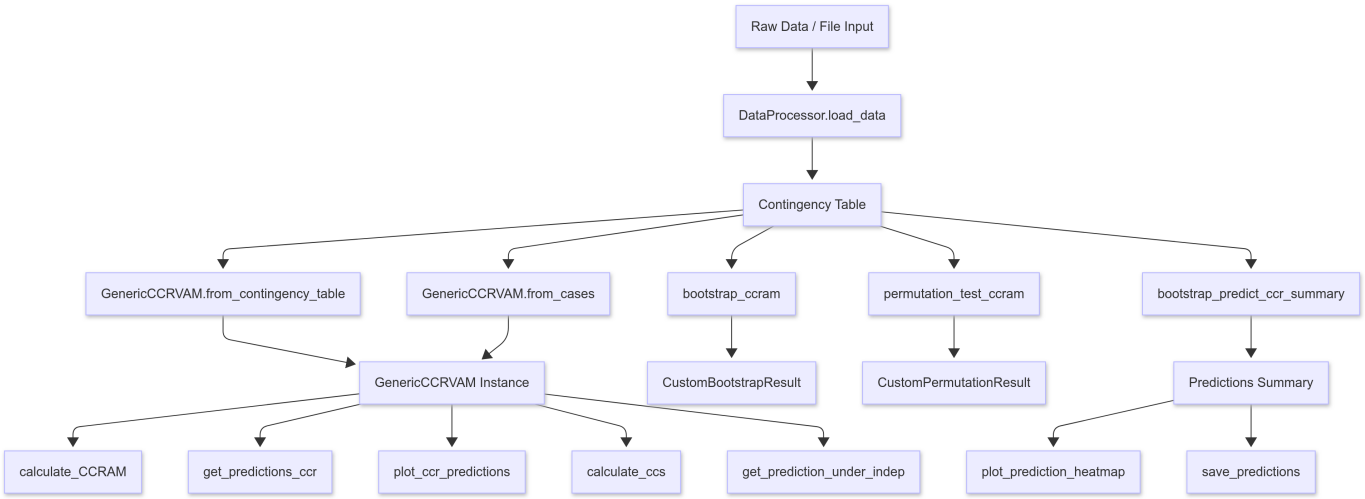


Figure 4.2: Functional Workflow for Checkerboard Copula Analysis

Figure 4.2 outlines the end-to-end computational pipeline within the `ccrvam` package. Raw categorical data is processed into a contingency table using the `DataProcessor`. This table can then be passed into the `GenericCCRVAM` class for association analysis, prediction, and visualization. Alternatively, the same input can be used in bootstrapping and permutation testing workflows to produce confidence intervals, p-values, and prediction heatmaps. The system supports both analytical modeling and robust statistical inference.

4.10.2 Vectorized Implementations and Error Handling

Performance optimization was a key consideration in the design, particularly for higher-dimensional tables. By leveraging NumPy's (Developers 2025) vectorized operations, Pandas's (Mckinney 2011) effective data handling, SciPy's (Virtanen et al. 2020) bootstrapping function calls, and Matplotlib's (Hunter 2007) efficient graphing APIs (Application Programming Interfaces) the package achieves significantly better performance than naive loop-based implementations, enabling analysis of larger datasets.

The package includes comprehensive input validation and error handling to provide informative

messages when issues arise. This approach helps users identify and fix problems quickly, improving the overall user experience. On the developer side, this allows for easy debugging and faster development of new features.

4.11 Testing, Validation, and Performance Evaluation

4.11.1 Comprehensive Test Suite

The `ccrvam` package includes a comprehensive test suite to ensure correctness and reliability across all implemented functionality. The test suite encompasses unit tests for all core functionality, verifying individual components in isolation. In contrast, integration tests confirm proper behavior in end-to-end workflows that simulate typical user interactions. Edge case testing rigorously examines boundary conditions where algorithms are most likely to fail, and dimensional-invariant testing validates consistent performance across varying complexity across 2D, 3D, and 4D contingency tables. The suite also incorporates regression tests to prevent reintroducing previously fixed bugs as the codebase evolves.

The package achieves over 93% code coverage, ensuring that most code paths and user experiences are well-tested and safe for production use in statistical analysis. In order to achieve better observability in our code and maintain a check ensuring that our tests pass irrespective of machine environments, we leveraged pytest (Krekel 2025) and coverage (Batchelder and Coverage.py 2025) Python libraries as our testing infrastructure. These tools provide a robust framework for automated test execution and detailed reporting on test coverage. An example test from our suite is shown below:

```
@pytest.mark.parametrize(
  "predictors, response, expected_sccram", [
    ([1], 2, 0.84375/(12*0.0703125)), # Single axis X1->X2
    ([2], 1, 0.0), # Single axis X2->X1
  ]
```

```
)
def test_calculate_SCCRAM(
  generic_ccrvam, predictors,
  response, expected_sccram
):
    """Test SCCRAM calculations with multiple conditioning axes."""
    calculated = generic_ccrvam.calculate_CCRAM(
                    predictors, response, scaled=True
                )
    np.testing.assert_almost_equal(calculated, expected_sccram)
```

Additional tests covering various aspects of the package's functionality can be found in the GitHub repository at ccrvam/tests.

4.11.2 Continuous Integration (CI)

To ensure code reliability and maintain compatibility across environments, the development workflow for `ccrvam` incorporates automated Continuous Integration (CI) using **GitHub Actions**. The CI pipeline is configured via `.yml` files to trigger test suites automatically upon every commit or pull request.

This setup enables testing across multiple Python versions—specifically 3.8, 3.9, 3.10, 3.11, 3.12, and 3.13—ensuring that the package remains robust and backward-compatible with older versions while embracing newer releases. Each test run verifies the integrity of core functionalities, including CCRAM/SCCRAM computations, bootstrapping procedures, and parallelization logic, helping to prevent regressions and promote long-term maintainability.

By integrating CI early in the development cycle, the package maintains high reliability and reproducibility throughout its evolution.

4.11.3 Performance Benchmarking

To ensure scalability and responsiveness when working with complex, high-dimensional contingency tables, the `ccrvam` package incorporates a series of optimizations for computational efficiency and

usability. Three core strategies are employed: (1) **vectorized implementations** for all core computations to eliminate Python-level loops; (2) **caching of intermediate statistics**, such as conditional distributions, to avoid redundant recalculations; and (3) **sparse-aware data structures**, which reduce memory overhead and accelerate access for large and sparse tables.

For example, in a contingency table of shape $2\times3\times2\times6$ (112 total cells), both the calculation of the Scaled CCRAM (SCCRAM) metric and the generation of bootstrap-based confidence intervals typically complete in under a few seconds on a modern CPU. This efficiency enables near real-time analysis of moderately sized categorical datasets.

A key component of the `ccrvam` package is the `bootstrap_predict_ccr_summary()` function, which estimates predictive performance across all combinations of predictor values via a bootstrap resampling procedure. This function computes a summary matrix of prediction percentages, showing the proportion of bootstrap samples assigned to each possible response category for each predictor combination. Notably, this method is designed with **native support for parallel execution**. By simply setting the `parallel=True` flag (enabled by default), users can leverage multiple CPU cores to accelerate computation. Internally, `ccrvam` utilizes Python's `concurrent.futures.ProcessPoolExecutor` to automatically detect and allocate available cores, executing bootstrap iterations independently and concurrently. Empirical benchmarks demonstrate the efficiency of this design: enabling parallelism yields a **3× reduction in runtime latency with 7 threads**, and up to an **15× speedup using 63 cores on FrostByte**, Amherst College's High Performance Computing (HPC) cluster. This substantial performance gain makes `ccrvam` highly scalable and well-suited for large-scale studies requiring intensive resampling procedures.

Finally, to enhance computational performance on larger datasets, `ccrvam` is implemented in such a way to allow for seamless GPU integration on the user side at the moment. By leveraging CuPy, a

NumPy-compatible library for GPU-accelerated computing (Okuta et al. 2017), users can offload core computations to NVIDIA CUDA or AMD ROCm GPUs with minimal code modifications. This enables significant speedups in parallelizable operations such as bootstrapping and marginalization over large multi-way tables.

4.12 User Documentation and Example Workflows

The `ccrvam` package includes comprehensive documentation and example workflows to help users get started:

4.12.1 API Documentation

The package provides detailed API documentation for all user and developer-facing functions and classes through Sphinx (Turner 2025). The documentation is hosted on ReadTheDocs at ccrvam.readthedocs.io. It includes function signatures, input argument descriptions, outputs documentation, warnings/errors log, usage examples on 2D and 4D sample datasets, and cross-references to related functions.

4.12.2 Example Workflows

The package includes example workflows to demonstrate common analysis patterns. These examples (located at ccrvam/examples/jupyter) demonstrate complete analysis workflows from data loading to visualization and statistical testing, helping users understand how to apply the package to their own research questions.

In the next chapter, we will use our `ccrvam` package to perform EDA on some real-world datasets.

Chapter 5

Real Data Analysis

This chapter demonstrates the practical application of the Checkerboard Copula Regression-based Visualization and Association Measure (CCRVAM) techniques introduced in previous chapters. We will analyze a real-world dataset using the `ccrvam` package implementation described in Chapter 4. This analysis will showcase how our methods can be used to explore associations between categorical predictors and an ordinal response variable, quantify the strength of these associations, and visualize prediction patterns.

5.1 Dataset Overview

The dataset we analyze contains information from a clinical study on back pain treatments, initially presented by Anderson (1984). This dataset is particularly suitable for our methodology as it includes an ordinal response variable (pain relief outcome) and multiple categorical predictor variables.

The dataset consists of 4 categorical variables:

Variable	Description	Categories
X_1	Length of Previous Attack	1=Short, 2=Long
X_2	Pain Change	1=Better, 2=Same, 3=Worse
X_3	Lordosis	1=Absent/Decreasing, 2=Present/Increasing
$Pain$	Back Pain Outcome	1=worse (W), 2=same (S), 3=slight improvement (SI), 4=moderate improvement (MODI), 5=marked improvement (MARI), 6=complete relief (CR)

This dataset represents a common scenario in medical and social science research, where the goal is to understand how multiple categorical factors influence an ordinal outcome. The pain outcome variable has a natural ordering (from worse to complete relief), making it an ideal candidate for our checkerboard copula approach.

5.2 Data Preparation and Loading

The `ccrvam` package provides flexible data loading capabilities through the `DataProcessor` class. As described in Chapter 4, this class supports multiple data formats, including case-form, frequency-form, and table-form data. Here, we demonstrate how to load the back pain dataset using each approach.

First, we need to import the necessary libraries and define our variable structure:

```
import numpy as np
from ccrvam import GenericCCRVAM, DataProcessor

# Define the ordered list of variable names
```

```
var_list_4d = ["x1", "x2", "x3", "pain"]

# Define the dimension tuple representing
# the number of categories for each variable
data_dimension = (2, 3, 2, 6)

# Create a category mapping for non-integer categories
# (required for 'pain' variable)
category_map_4d = {
    "pain": {
        "worse": 1,
        "same": 2,
        "slight.improvement": 3,
        "moderate.improvement": 4,
        "marked.improvement": 5,
        "complete.relief": 6
    },
}
```

The `var_list_4d` defines the order of variables in our analysis. The `data_dimension` tuple specifies the number of categories for each variable in the same order. The `category_map_4d` maps text labels to numeric indices for non-integer categories, which is necessary for this dataset's "pain" variable.

The `ccrvam` package supports three different data loading formats, providing flexibility based on how your data is structured. We will demonstrate each method:

5.2.1 Case Form Data Loading

Case form represents individual observations, where each row contains the category values for all variables for a single observation:

```
# Loading data from the case form file
contingency_table_4d = DataProcessor.load_data(
    "./data/caseform.pain.txt",
    data_form="case_form",
    dimension=data_dimension,
    var_list=var_list_4d,
    category_map=category_map_4d,
    named=True,
    delimiter="\t"
)

```

```
# Initialize the GenericCCRVAM object
rda_ccrvam = GenericCCRVAM.from_contingency_table(
                contingency_table_4d
             )
```

5.2.2 Frequency Form Data Loading

Frequency form data contains the category values for all variables along with a count of how many times that combination appears:

```
# Loading data from the frequency form file
contingency_table_4d_from_freq = DataProcessor.load_data(
    "./data/freqform.pain.txt",
    data_form="frequency_form",
    dimension=data_dimension,
    var_list=var_list_4d,
    category_map=category_map_4d,
    named=True,
    delimiter="\t"
)

# Initialize the GenericCCRVAM object
rda_ccrvam_from_freq = GenericCCRVAM.from_contingency_table(
                           contingency_table_4d_from_freq
                       )
```

5.2.3 Contingency Table Form Data Loading

Table form represents the data as a multidimensional contingency table with counts directly:

```
# Define the 4D contingency table as a NumPy array
rda_contingency_table = np.array([
    # X1=1 (Short)
    [
        # X2=1 (Better)
        [
            # X3=1 (Absent)
            [0, 1, 0, 0, 2, 4],  # Counts for each Pain outcome
            # X3=2 (Present)
            [0, 0, 0, 1, 3, 0]
        ],
        # X2=2 (Same)
        [
```

```
            # X3=1 (Absent)
            [0, 2, 3, 0, 6, 4],
            # X3=2 (Present)
            [0, 1, 0, 2, 0, 1]
        ],
        # X2=3 (Worse)
        [
            # X3=1 (Absent)
            [0, 0, 0, 0, 2, 2],
            # X3=2 (Present)
            [0, 0, 1, 1, 3, 0]
        ]
    ],
    # X1=2 (Long)
    [
        # X2=1 (Better)
        [
            # X3=1 (Absent)
            [0, 0, 3, 0, 1, 2],
            # X3=2 (Present)
            [0, 1, 0, 0, 3, 0]
        ],
        # X2=2 (Same)
        [
            # X3=1 (Absent)
            [0, 3, 4, 5, 6, 2],
            # X3=2 (Present)
            [1, 4, 4, 3, 0, 1]
        ],
        # X2=3 (Worse)
        [
            # X3=1 (Absent)
            [2, 2, 1, 5, 2, 0],
            # X3=2 (Present)
            [2, 0, 2, 3, 0, 0]
        ]
    ]
])

# Load data from the table
contingency_table_4d_from_array = DataProcessor.load_data(
    rda_contingency_table,
    data_form="table_form",
    dimension=data_dimension,
    var_list=var_list_4d,
    category_map=category_map_4d
)

# Initialize the GenericCCRVAM object
rda_ccrvam_from_array = GenericCCRVAM.from_contingency_table(
                          contingency_table_4d_from_array
                        )
```

```python
# Check if the Resulting Joint Probability Matrices are the same
# after loading data using various methods mentioned above
same_1_2 = np.array_equal(rda_ccrvam.P, rda_ccrvam_from_freq.P)
same_2_3 = np.array_equal(rda_ccrvam_from_freq.P, rda_ccrvam_from_array.P)

# Are P matrices the same across methods?
print(same_1_2 and same_2_3)
```

```
True
```

The output of each loading method is a 4-dimensional joint probability matrix with shape $(2, 3, 2, 6)$ corresponding to the number of categories for each variable. This matrix contains the estimated joint probability distribution for all possible combinations of the categorical variables.

The values in the matrix represent the probability of observing each specific combination of categories. For example, the value at position $[0, 0, 0, 1] = 0.0099$ represents the probability of observing: $X_1 = 1$ (Short previous attack), $X_2 = 1$ (Better pain change), $X_3 = 1$ (Absent/Decreasing Lordosis), and $Pain = 2$ (Same pain outcome).

Note that all three loading methods should produce the same joint probability matrix if the data sources are consistent, which we can observe from the identical outputs in the example.

> **Note:**
>
> For brevity, we will not be walking through each code-chunk for further real data analysis covered in this chapter. If you are interested, please feel free to check out the code for the real data analysis in the Jupyter notebooks at ccrvam/examples/jupyter.

5.3 Exploratory Data Analysis

Before applying our advanced statistical methods, we examine the data's basic probability distributions. These distributions provide insights into the prevalence of each category in our dataset. We

observe the following marginal probability density functions (pdfs):

- **Length of Previous Attack (X_1)**: 38.61% of patients had short previous attacks, while 61.39% had long previous attacks.
- **Pain Change (X_2)**: 20.79% of patients experienced better pain change, 51.49% had the same pain level, and 27.72% experienced worse pain change.
- **Lordosis (X_3)**: 63.37% of patients had absent or decreasing Lordosis, while 36.63% had present or increasing Lordosis.
- **Back Pain Outcome ($Pain$)**: The distribution shows that 4.95% of patients experienced worse pain after treatment (W), 13.86% reported no change (S), 17.82% experienced slight improvement (SI), 19.80% had moderate improvement (MODI), 27.72% reported marked improvement (MARI), and 15.84% experienced complete relief (CR).

These findings provide valuable context for interpreting our subsequent analyses. For the back pain outcome specifically, we observe that treatments were generally effective, with more than 60% of patients experiencing at least moderate improvement (combining the moderate improvement, marked improvement, and complete relief categories).

5.4 Calculating Checkerboard Copula Scores (CCS)

Following the methodology described in Chapter 3, we calculate the checkerboard copula scores (CCS) for each variable in our dataset. We compute the CCS for all variables and determine their respective variances:

- **Length of Previous Attack (X_1):** Two distinct scores (0.193, 0.693) with a variance of 0.059.
- **Pain Change (X_2):** Three distinct scores (0.104, 0.465, 0.861) with a variance of 0.069.
- **Lordosis (X_3):** Two distinct scores (0.317, 0.817) with a variance of 0.058.

- **Back Pain Outcome (*Pain*):** Six distinct scores ranging from 0.025 to 0.921, with a variance of 0.080.

The results show distinct patterns of scores across the variables, with Back Pain Outcome demonstrating the most granular distribution with six distinct scores ranging from approximately 0.025 to 0.921. The variance calculations reveal that the Back Pain Outcome variable has the highest variance at approximately 0.080, while Lordosis (X_3) has the lowest at 0.058. These variance values are critical inputs for our subsequent analysis of scaled association measures, as they provide normalization factors that allow for meaningful comparisons across different variable relationships.

5.5 Checkerboard Copula Regression (CCR) Analysis

Next, we apply the Checkerboard Copula Regression (CCR) to predict the back pain outcome categories based on the predictor variables: Length of Previous Attack (X_1), Pain Change (X_2), and Lordosis (X_3).

Our analysis generates predictions for each possible combination of predictor variables. The predictions show the expected pain outcome category for each combination of predictor variables. For example, patients with a short previous attack ($X_1 = 1$), better pain change ($X_2 = 1$), and absent Lordosis ($X_3 = 1$) are predicted to have marked improvement (category 5) in pain outcomes. The results reveal several important patterns:

1. Patients with a short previous attack ($X_1 = 1$) generally have better outcomes (categories 4-5) than those with long previous attacks ($X_1 = 2$)

2. Within the short previous attack group, those with better or worse pain change ($X_2 = 1$ or $X_2 = 3$) tend to have marked improvement (category 5)

3. For patients with long previous attacks ($X_1 = 2$), those with worse pain change ($X_2 = 3$) generally have the poorest outcomes (category 3: slight improvement)

4. Under the assumption of joint independence between *Pain* and the predictor variables (X_1, X_2, X_3), the predicted pain outcome category is 4 (moderate improvement), which serves as a reference point for our analysis.

These findings highlight the complex interrelationships between previous attack duration, pain change, and Lordosis in predicting back pain treatment outcomes. The visualizations generated by `ccrvam` in Figure 5.1 further enhance our understanding of these relationships and provide valuable clinical insights.

5.6 Quantifying Association with (S)CCRAM

We now quantify the strength of the association between our predictors (Length of Previous Attack, Pain Change, and Lordosis) and the back pain outcome using CCRAM and SCCRAM. The CCRAM (Checkerboard Copula Regression Association Measure) value of 0.2576 indicates that the three predictor variables can explain approximately 25.76% of the variation in back pain outcomes. This provides a meaningful measurement of how well our predictor variables collectively explain the pain outcomes observed in patients.

We calculate the SCCRAM (Scaled Checkerboard Copula Regression Association Measure) for a more standardized interpretation, which yields a value of 0.2687. This scaled measure accounts for the theoretical maximum association possible in this dataset structure, making it easier to interpret and compare across different studies with varying data characteristics. These association measures provide important quantitative validation of the relationships we observed in our earlier analyses and help establish the overall predictive power of our model.

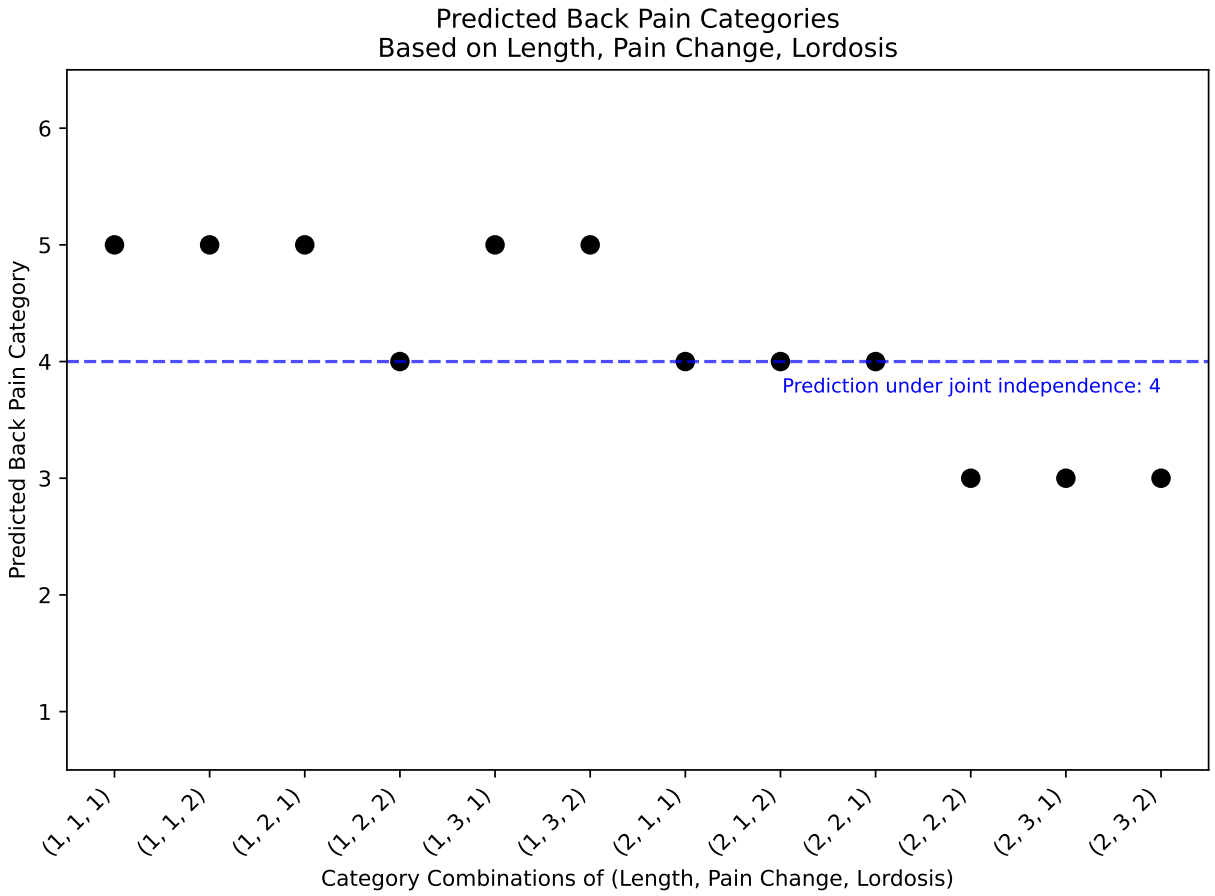


Figure 5.1: This visualization illustrates the predicted back pain outcomes based on combinations of prior attack length, pain change, and lordosis severity using the ccrvam framework. Each tuple on the x-axis represents a unique combination of predictor categories, with the predicted back pain category indicated by the position of the black dot.

5.7 Uncertainty Quantification Using Bootstrap

To assess the uncertainty in our CCRAM and SCCRAM estimates, we utilize nonparametric bootstrap methods with 9,999 resamples. This approach allows us to estimate confidence intervals and standard errors without making distributional assumptions about our data.

For the CCRAM measure quantifying the association between our predictors (Length of Previous Attack, Pain Change, and Lordosis) and Back Pain outcomes, the bootstrap analysis yields an observed CCRAM value of 0.2576 with a 95% BCa confidence interval of $(0.1849, 0.4762)$ and a standard error of 0.0748. For the scaled measure (SCCRAM), which normalizes the association for better interpretability, we observe a value of 0.2687 with a 95% BCa confidence interval of $(0.0691, 0.3509)$ and a standard error of 0.0775.

These results reveal important insights about our analysis. The confidence intervals indicate that while there is uncertainty in the exact value of the association, we can be reasonably confident that the true association is substantial. The positive bias in the bootstrap estimates (0.0666 for CCRAM and 0.0718 for SCCRAM) suggests that our observed values may be conservative estimates of the true association. The relatively high bias-to-standard-error ratios (0.8904 for CCRAM and 0.9267 for SCCRAM) indicate some potential complexity in the underlying distribution, which further justifies our use of robust bootstrap methods for uncertainty quantification.

The visualizations generated by `ccrvam` in Figure 5.2 and Figure 5.3 further enhance our understanding of the uncertainty in (S)CCRAM by providing an intuitive representation.

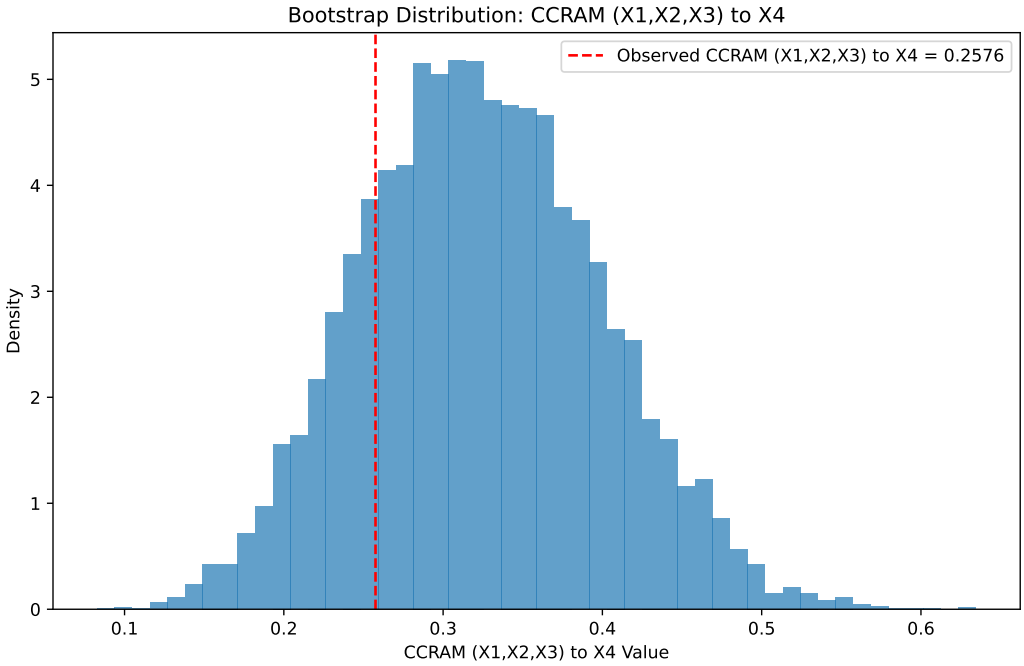


Figure 5.2: Bootstrap distribution of CCRAM $(X_1, X_2, X_3 \rightarrow X_4)$. The red dashed line marks the observed CCRAM value of 0.2576. This plot visualizes variability and supports estimation of confidence intervals and bias for the measure of association.

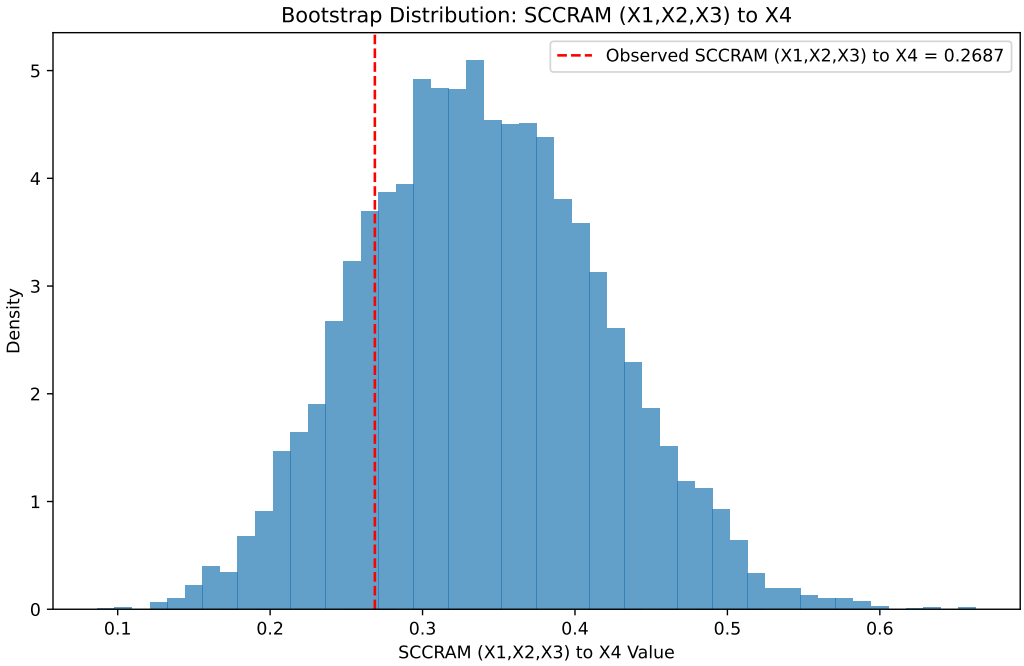


Figure 5.3: Bootstrap distribution of SCCRAM $(X_1, X_2, X_3 \rightarrow X_4)$. The red dashed line marks the observed SCCRAM value of 0.2687. This normalized version of CCRAM accounts for maximum possible association and highlights uncertainty in scaled estimates.

5.8 Statistical Significant Testing Using Permutation Tests

To assess whether the observed associations could have occurred by chance, we conducted permutation tests with 9,999 resamples. This approach allows us to construct empirical null distributions for both CCRAM and SCCRAM metrics under the hypothesis of no association between predictors and the pain outcome.

For the CCRAM measure, we observe a value of 0.2576 with a p-value of 0.0016. The permutation distribution exhibits a median of 0.0998, with the 99th percentile at 0.2214. Our observed CCRAM value exceeds even the 99th percentile of the null distribution, providing strong evidence against the null hypothesis of no association. Similarly, for the SCCRAM measure, we observe a value of 0.2687 with an even smaller p-value of 0.0011. The permutation distribution for SCCRAM shows a median of 0.1046, with the 99th percentile at 0.2255. Again, our observed value exceeds the 99th percentile of values that would be expected by chance.

These permutation test results provide strong statistical evidence that the observed associations between our predictor variables (Length of Previous Attack, Pain Change, and Lordosis) and Back Pain outcomes are not due to random variation. The extremely small p-values confirm that these relationships are statistically significant, further validating the clinical relevance of our findings.

The visualizations generated by `ccrvam` in Figure 5.4 and Figure 5.5 further enhance our understanding of the relative association strength in (S)CCRAM by providing an intuitive representation.

5.9 Bootstrap Analysis for CCR Predictions

We can also use bootstrap methods to assess the uncertainty in our category predictions. By generating 9,999 bootstrap samples, we obtain a prediction matrix that shows the percentage of bootstrap samples

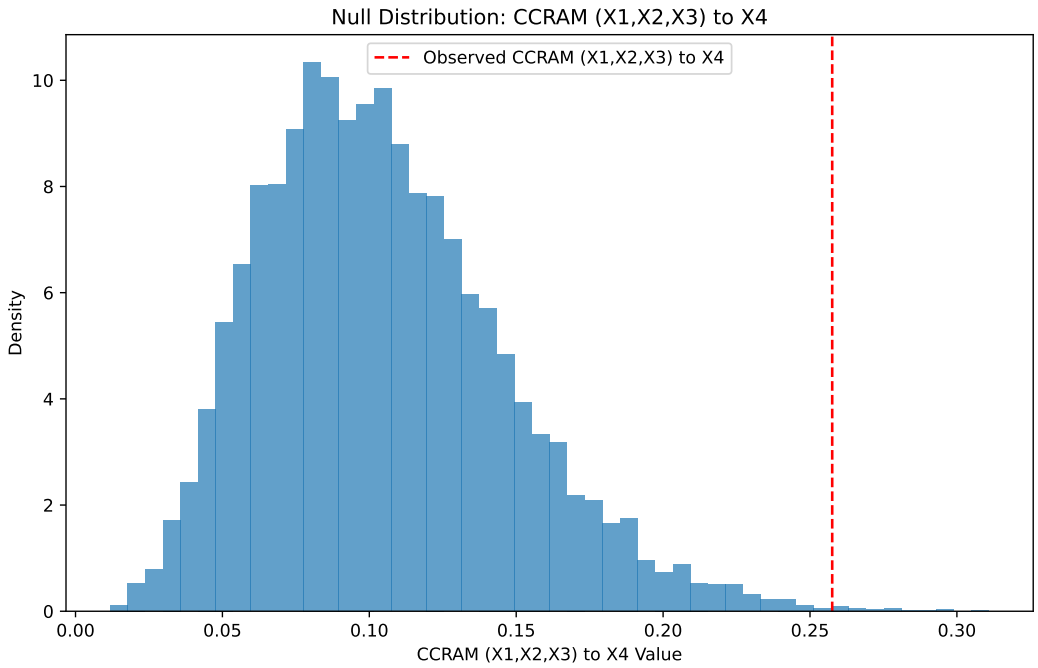


Figure 5.4: Permutation distribution of CCRAM $(X_1, X_2, X_3 \to X_4)$. The red dashed line marks the observed CCRAM value of 0.2576. The empirical null distribution illustrates that such a value is highly unlikely under the assumption of no association (p = 0.0016).

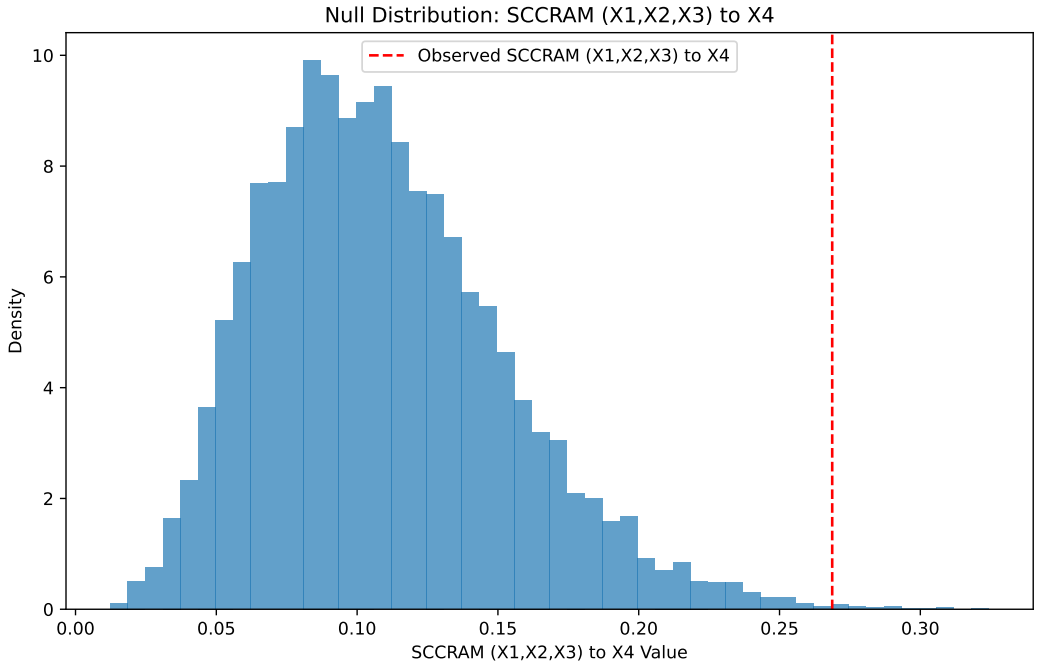


Figure 5.5: Permutation distribution of SCCRAM ($X_1, X_2, X_3 \rightarrow X_4$). The red dashed line marks the observed SCCRAM value of 0.2687. The observed value lies well beyond the 99th percentile of the null distribution, providing strong evidence of statistically significant association (p = 0.0011).

predicting each pain category for each combination of predictor values.

This approach provides a measure of confidence in our predictions. For example, for patients with short previous attack, better pain change, and absent Lordosis ($X_1 = 1, X_2 = 1, X_3 = 1$), the prediction of category 5 (marked improvement) occurs in approximately 70.90% of bootstrap samples, indicating high confidence in this prediction. Similarly, for patients with short previous attack, better pain change, and present Lordosis ($X_1 = 1, X_2 = 1, X_3 = 2$), the prediction of category 5 is even more consistent, occurring in 87.17% of bootstrap samples.

The bootstrap analysis also reveals cases where predictions are less certain. For instance, patients with a long previous attack, the same pain change, and present Lordosis ($X_1 = 2, X_2 = 2, X_3 = 2$) show 76.92% of bootstrap samples predicting category 3 (slight improvement). In contrast, patients with long previous attack, better pain change, and present Lordosis ($X_1 = 2, X_2 = 1, X_3 = 2$) have more uncertainty, with 52.06% of samples predicting category 5 and 36.29% predicting category 4.

Visualizing this bootstrap prediction matrix through a heatmap produced by `ccrvam` in Figure 5.6 further enhances our understanding of prediction confidence across different combinations of predictor variables. The dotted line in the heatmap indicates the predicted category under joint independence (category 4), providing a reference point against which to compare our model predictions.

5.10 Discussion and Clinical Interpretation

Our analysis of the back pain treatment dataset using the Checkerboard Copula Regression-based Visualization and Association Measure (CCRVAM) methodology reveals several clinically meaningful patterns that merit discussion. The CCRAM value of 0.2576 indicates that approximately 25.76% of the variation in back pain outcomes can be explained by the three predictor variables (Length of Previous Attack, Pain Change, and Lordosis). Similarly, the SCCRAM value of 0.2687 provides a

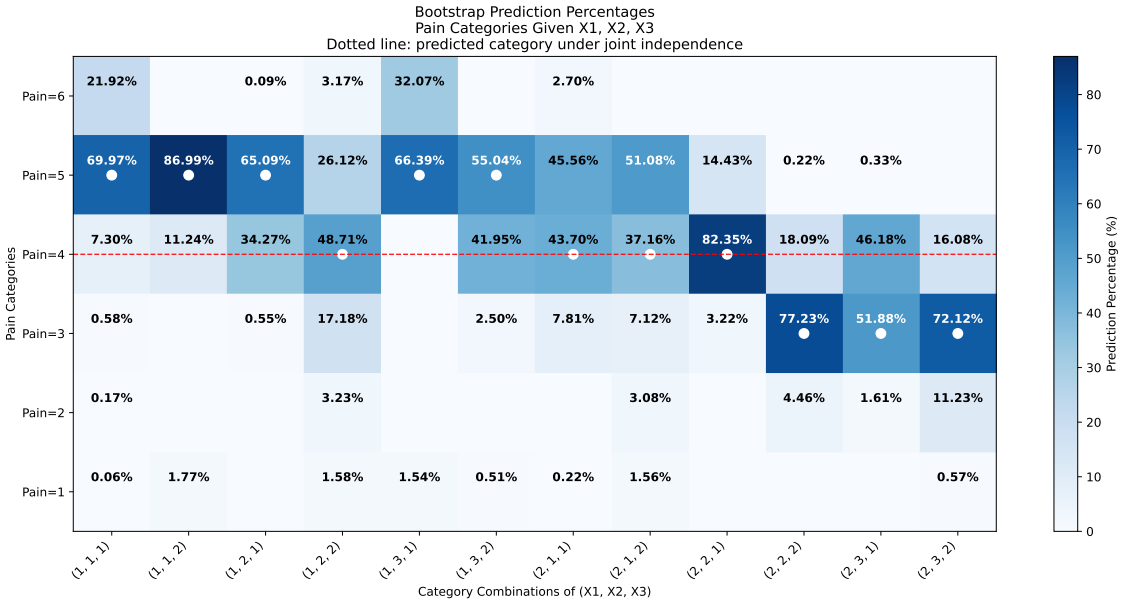


Figure 5.6: Bootstrap-based heatmap of predicted pain categories given (X_1, X_2, X_3). Each cell shows the percentage of bootstrap samples predicting a given pain level for a specific combination of predictor values. White dots mark the most frequently predicted category, and the red dotted line indicates the expected response under joint independence.

standardized measure that accounts for the theoretical maximum association possible in this dataset structure. These values demonstrate a moderate but meaningful association between our predictors and pain outcomes, consistent with the multifactorial nature of back pain treatment response.

Patients with short previous attacks ($X_1 = 1$) generally experienced better outcomes (categories 4-5: moderate to marked improvement) compared to those with long previous attacks ($X_1 = 2$). This finding suggests that the chronicity of pain prior to treatment may be an important prognostic factor, with early intervention potentially yielding better results. The influence of pain change (X_2) appears to interact with the length of the previous attack. For patients with short previous attacks, both better ($X_2 = 1$) and worse ($X_2 = 3$) pain change categories often led to marked improvement (category 5), while for patients with long previous attacks, worse pain change ($X_2 = 3$) generally predicted poorer outcomes (category 3: slight improvement). This interaction effect highlights the complex nature of pain response trajectories. The presence or absence of Lordosis (X_3) appears to have a more subtle

influence on outcomes compared to the other predictors, often modifying the effects of the primary predictors rather than driving outcomes independently.

The permutation test results provide strong statistical evidence that the observed associations are not due to random variation. With p-values of 0.0016 for CCRAM and 0.0011 for SCCRAM, we can confidently reject the null hypothesis of no association between our predictor variables and back pain outcomes. The observed values exceed the 99th percentile of their respective null distributions, further strengthening the significance of our findings.

Our predictions' bootstrap analysis reveals varying proportions across different predictor combinations. For instance, for patients with short previous attacks, better pain change, and absent Lordosis ($X_1 = 1, X_2 = 1, X_3 = 1$), marked improvement (category 5) is predicted in 69.97% of bootstrap samples. For patients with short previous attacks, better pain change, and present Lordosis ($X_1 = 1, X_2 = 1, X_3 = 2$), this prediction is even more consistent, occurring in 86.99% of bootstrap samples. In contrast, patients with long previous attack, better pain change, and present Lordosis ($X_1 = 2, X_2 = 1, X_3 = 2$) show more uncertainty, with 51.08% of samples predicting category 5 and 37.16% predicting category 4. These confidence metrics provide valuable context for clinical decision-making, indicating where predictions are most reliable and greater caution may be warranted.

These findings have several implications for clinical practice. Clinicians may use these results to provide more informed prognostic guidance to patients based on their specific combination of risk factors. The identification of patients with long previous attacks and worsening pain as having poorer outcomes may suggest the need for more aggressive or multimodal intervention approaches for this subgroup. The generally better outcomes observed in patients with shorter previous attacks reinforce the importance of early treatment initiation for back pain. Additionally, the bootstrap prediction matrices can help calibrate expectations for both clinicians and patients, providing nuanced

probability estimates rather than deterministic predictions.

This analysis demonstrates several key advantages of the CCRVAM methodology for analyzing categorical and ordinal data in clinical research. The approach respects the ordinal nature of predictors and response variables without imposing arbitrary numerical scoring. CCRVAM naturally captures complex interactions among predictors without requiring explicit interaction terms. The visualization tools the `ccrvam` package provides facilitate the intuitive interpretation of multidimensional patterns. The integration of bootstrap and permutation methods provides a comprehensive uncertainty assessment without parametric assumptions.

Despite these strengths, several limitations should be acknowledged. The dataset includes 101 observations distributed across 72 possible predictor and outcome category combinations, resulting in sparse data for some combinations. As with any observational study, unmeasured confounders may influence the observed associations. The cross-sectional nature of the data limits our ability to assess temporal relationships and treatment dynamics.

The application of CCRVAM methodology to this back pain treatment dataset using our `ccrvam` package has yielded clinically meaningful insights while demonstrating the utility of this approach for analyzing complex categorical data in medical research. The findings suggest that patient characteristics significantly influence treatment outcomes, particularly the length of previous pain episodes and early treatment response. These results may inform more personalized approaches to back pain management and highlight the value of sophisticated methodological tools for extracting meaningful patterns from categorical clinical data.

Chapter 6

Conclusion and Future Work

Exploratory Data Analysis of multivariate categorical data with ordinal responses presents unique challenges that traditional statistical methodologies often struggle to address effectively. This thesis has presented a comprehensive framework synthesizing copula theory, regression dependence concepts, and modern computational approaches to overcome these limitations. By starting with an introduction to dependence, exploring continuous copula, bridging to discrete cases, building upon the methodological innovations proposed by Wei and Kim (2021), and extending their practical implementation, we have developed a robust analytical workflow encapsulated in a novel Python package, providing researchers with accessible tools for rigorous analysis of complex categorical data structures. The primary contribution of this work lies in bridging the theoretical foundations of copula theory with practical exploratory data analysis, resulting in a unified framework tailored explicitly for categorical data analysis. The `ccrvam` Python package represents the first end-to-end implementation of Checkerboard Copula Regression methods, enabling researchers to apply these sophisticated techniques without requiring extensive knowledge of the underlying mathematical complexities. Furthermore, rigorous testing on both simulated and real-world datasets has demonstrated the robustness and utility of the CCR approach and (S)CCRAM measures for quantifying

regression dependence in multivariate categorical data.

Unlike traditional association measures that often treat variables symmetrically, the CCR approach explicitly accommodates regression dependence, preserving the crucial distinction between response and explanatory variables. This orientation toward regression dependence makes the methods particularly valuable in fields such as healthcare, social sciences, and economics, where understanding causal and predictive relationships is paramount. The model-free nature of CCR and (S)CCRAM offers significant advantages over parametric approaches by avoiding potentially restrictive assumptions about functional forms, allowing researchers to explore dependencies in data without imposing predetermined structures. This flexibility reveals patterns that conventional modeling approaches might miss. Developing novel visualization methods enhances the interpretability of dependence structures in categorical data, going beyond traditional contingency table representations to provide intuitive visual insights into complex multivariate relationships.

Despite these advances, several limitations warrant acknowledgment in the current implementation. As the dimensionality of the contingency table increases, the computational requirements for CCR analysis grow substantially, potentially challenging the analysis of extremely high-dimensional datasets despite our implemented optimizations. While (S)CCRAM provides a quantitative measure of association strength, interpreting the practical significance of specific values remains more art than science, lacking the direct translation to marginal effects that traditional regression coefficients offer. As with many non-parametric methods, CCR approaches require adequate sample sizes to estimate dependence structures reliably, particularly for complex multivariate scenarios. The current implementation also requires complete data in contingency tables, with no built-in mechanisms for addressing missing values beyond the preprocessing steps.

Integrating bootstrap resampling and permutation testing within our framework provides much-

needed statistical inference capabilities previously unavailable for model-free approaches to categorical data analysis. These techniques enable researchers to quantify uncertainty in predictions and association measures, offering confidence intervals and formal hypothesis tests that enhance the rigor of categorical data analysis. The `ccrvam` package thus democratizes access to advanced statistical methods by abstracting away complex mathematical details while preserving methodological rigor, enabling researchers across disciplines to incorporate these techniques into their analytical workflows without specialized statistical expertise.

This work opens several promising avenues for future research and methodological development. One crucial direction involves formalizing conditional (S)CCRAM, as theorized in Wei et al. (2023), which would extend the utility of these dependence measures by enabling the quantification of partial associations while controlling for confounding variables. Another key extension lies in broadening the framework to support joint analysis of continuous and categorical variables, enhancing its applicability across a broader range of empirical settings. Similarly, adapting CCR-based techniques for longitudinal or time series categorical data would allow for studying temporal dependencies in ordinal outcomes.

From a computational standpoint, advancing algorithmic efficiency remains a critical priority, particularly for high-dimensional scenarios. Future work could explore sparse matrix representations, approximate inference, or other memory-efficient strategies. While this thesis introduces basic parallelization support, further integration with "embarrassingly parallel frameworks" and GPU-accelerated computation on the developer side (without user overhead) would significantly improve scalability—especially for the bootstrap and permutation procedures of (S)CCRAM, which are often the computationally intensive components of the analysis pipeline.

Further advances in visualization and explainability present another promising direction, with the potential for developing interactive visualization components that allow researchers to explore

dependence structures dynamically. Research into decomposing (S)CCRAM values into interpretable components could enhance the explanatory power of the measure. Domain-specific implementations tailored for healthcare, finance, or social sciences could enhance adoption by incorporating field-specific knowledge and constraints while developing bridges to common statistical environments to facilitate incorporation into established research pipelines.

Exploring regression dependence in multivariate categorical data remains a rich and challenging area of statistical research. As data collection continues to proliferate across fields, the need for sophisticated yet accessible methods for categorical data analysis will only increase. This thesis has contributed to advancing this field by reviewing theoretical insights and engineering novel practical tools through the lens of checkerboard copulas and associated regression methods. We aim to enhance researchers' ability to extract meaningful insights from categorical data across diverse domains by enabling model-free exploration of complex dependence structures. The framework presented in this thesis represents one step toward addressing this need. However, much work remains to realize the potential of copula-based approaches for exploratory data analysis of categorical information. We hope that both the methodological advances reviewed and the software tools presented here will serve as valuable resources for researchers and inspire continued innovation.

References

Anderson, J. A. (1984), "Regression and ordered categorical variables," *Journal of the Royal Statistical Society. Series B (Methodological)*, [Royal Statistical Society, Oxford University Press], 46, 1–30.

Anscombe, F. J. (1973), "Graphs in statistical analysis," *The American Statistician*, American Statistical Association, Taylor & Francis, Ltd., 27, 17–21.

Batchelder, N., and Coverage.py, C. to (2025), "Coverage.py: The code coverage tool for python," https://coverage.readthedocs.io/.

Buja, A., Cook, D., Hofmann, H., Lawrence, M., Lee, E.-K., Swayne, D. F., and Wickham, H. (2009), "Statistical inference for exploratory data analysis and model diagnostics," *Philosophical Transactions of the Royal Society of London*, 367, 4361–4383.

Davison, A. C., and Hinkley, D. V. (1997), *Bootstrap methods and their application*, Cambridge series in statistical and probabilistic mathematics, Cambridge: Cambridge University Press.

Denuit, M., and Lambert, P. (2005), "Constraints on concordance measures in bivariate discrete data," *Journal of Multivariate Analysis*, 93, 40–57.

Developers, N. (2025), "NumPy 2.2.4: Fundamental package for array computing in python," https://pypi.org/project/numpy/.

Donoho, D. L. (2017), "50 years of data science," *Journal of Computational and Graphical Statistics*, 26, 745–766.

Efron, B. (1987), "Better bootstrap confidence intervals," *Journal of the American Statistical Association*, 82, 171–185. https://doi.org/10.2307/2289144.

Erdely, A. (2017), "A subcopula based dependence measure," *Kybernetika*, Institute of Information Theory; Automation AS CR, 53, 231–243.

Faugeras, O. P. (2017), *Dependence Modeling*, 5, 121–132. https://doi.org/doi:10.1515/demo-2017-0008.

Geenens, G. (2020), "Copula modeling for discrete random vectors," *Dependence Modeling*, 8, 417–440. https://doi.org/doi:10.1515/demo-2020-0022.

Gelman, A., and Vehtari, A. (2021), "What are the most important statistical ideas of the past 50 years?" *Journal of the American Statistical Association*, 116, 2087–2097.

Genest, C., Nešlehová, J. G., and Rémillard, B. (2014), "On the empirical multilinear copula process for count data," *Bernoulli*, 20, 1344–1371.

Genest, C., Nešlehová, J., and Remillard, B. (2017), "Asymptotic behavior of the empirical multilinear copula process under broad conditions," *Journal of Multivariate Analysis*, 159. https://doi.org/10.1016/j.jmva.2017.04.002.

Hesterberg, T. (2014), "What teachers should know about the bootstrap: Resampling in the undergraduate statistics curriculum."

Hofert, M., Kojadinovic, I., Maechler, M., and Yan, J. (2018), *Elements of Copula Modeling with r*, Springer Use R! Series.

Hunter, J. D. (2007), "Matplotlib: A 2D graphics environment," *Computing in Science & Engineering*, IEEE COMPUTER SOC, 9, 90–95. https://doi.org/10.1109/MCSE.2007.55.

Joe, H. (2014), *Dependence modeling with copulas*, Chapman; Hall/CRC, New York.

Krekel, H. (2025), "Pytest 8.3.5: Simple powerful testing with python," https://pypi.org/project/pytest/.

Liao, S.-M., Wang, L., and Kim, D. (2024), "Visualization of dependence in multidimensional contingency tables with an ordinal dependent variable via copula regression," in *Dependent data in social sciences research: Forms, issues, and methods of analysis*, eds. M. Stemmler, W. Wiedermann, and F. L. Huang, Cham: Springer International Publishing, pp. 517–538. https://doi.org/10.1007/978-3-031-56318-8_21.

Mckinney, W. (2011), "Pandas: A foundational python library for data analysis and statistics," *Python High Performance Science Computer*.

Nelsen, R. B. (2006), *An introduction to copulas*, Springer Science & business media.

Nešlehová, J. (2007), "On rank correlation measures for non-continuous random variables," *Journal of Multivariate Analysis*, 98, 544–567.

Okuta, R., Unno, Y., Nishino, D., Hido, S., and Loomis, C. (2017), "CuPy: A NumPy-compatible library for NVIDIA GPU calculations," in *Proceedings of workshop on machine learning systems (LearningSys) in the thirty-first annual conference on neural information processing systems (NIPS)*.

Rüschendorf, L. (2009), "On the distributional transform, sklar's theorem, and the empirical copula process," *Journal of Statistical Planning and Inference*, 139, 3921–3927. https://doi.org/10.1016/j.jspi.2009.05.030.

Schweizer, B., and Sklar, A. (1974), "Operations on distribution functions not derivable from operations on random variables," *Studia Mathematica*, 52, 43–52.

Shearer, C. (2000), "The CRISP-DM model: The new blueprint for data mining," *Journal of Data Warehousing*, 5, 13–22.

Sklar, M. (1959), "Fonctions de repartition an dimensions et leurs marges," *Publ. inst. statist. univ. Paris*, 8, 229–231.

Sveidqvist, K., and Mermaid, C. to (2014), *Mermaid: Generate diagrams from markdown-like text*, https://mermaid.js.org/.

Tufte, E. R. (1983), *The visual display of quantitative information*, Cheshire, CT: Graphics Press.

Tukey, J. W. (1977), *Exploratory data analysis*, AddisonWesley; Boston, MA.

Turner, A. (2025), "Sphinx 8.2.3: Python documentation generator," https://pypi.org/project/Sphinx/.

Ushey, K., and Wickham, H. (2024), *Renv: Project environments*.

VG, C. (2024), "Pypi-template 0.8.0: Template-based common/best practices for managing a python package on PyPi," https://pypi.org/project/pypi-template/.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., Walt, S. J. van der, Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., Carey, C. J., Polat, İ., Feng, Y., Moore, E. W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E. A., Harris, C. R., Archibald, A. M., Ribeiro, A. H., Pedregosa, F., Mulbregt, P. van, Vijaykumar, A., Bardelli, A. P., Rothberg, A., Hilboll, A., Kloeckner, A., Scopatz, A., Lee, A., Rokem, A., Woods, C. N., Fulton, C., Masson, C., Häggström, C., Fitzgerald, C., Nicholson, D. A., Hagen, D. R., Pasechnik, D. V., Olivetti, E., Martin, E., Wieser, E., Silva, F., Lenders, F., Wilhelm, F., Young, G., Price, G. A., Ingold, G.-L., Allen, G. E., Lee, G. R., Audren, H., Probst, I., Dietrich, J. P., Silterra, J., Webber, J. T., Slavič, J., Nothman, J., Buchner, J., Kulick, J., Schönberger, J. L., Miranda Cardoso, J. V. de, Reimer, J., Harrington, J., Rodríguez, J. L. C., Nunez-Iglesias, J., Kuczynski, J., Tritz, K., Thoma, M., Newville, M., Kümmerer, M., Bolingbroke, M., Tartre, M., Pak, M., Smith, N. J., Nowaczyk, N., Shebanov, N., Pavlyk, O., Brodtkorb, P. A., Lee, P., McGibbon, R. T., Feldbauer, R., Lewis, S., Tygier, S., Sievert, S., Vigna, S., Peterson, S., More, S., Pudlik, T., Oshima, T., Pingel, T. J., Robitaille, T. P., Spura, T., Jones, T. R., Cera, T., Leslie, T., Zito, T., Krauss, T., Upadhyay, U., Halchenko, Y. O., and Vázquez-Baeza, Y. (2020), "SciPy 1.0: Fundamental algorithms for scientific computing in python," *Nature Methods*, Springer Science; Business Media LLC, 17, 261–272.

https://doi.org/10.1038/s41592-019-0686-2.

Wei, Z., and Kim, D. (2021), "On exploratory analytic method for multi-way contingency tables with an ordinal response variable and categorical explanatory variables," *Journal of Multivariate Analysis*, 186, 104793. https://doi.org/10.1016/j.jmva.2021.104793.

Wei, Z., Wang, L., Liao, S.-M., and Kim, D. (2023), "On the exploration of regression dependence structures in multidimensional contingency tables with ordinal response variables," *Journal of Multivariate Analysis*, 196, 105179. https://doi.org/10.1016/j.jmva.2023.105179.

Appendix A

Code availability

This thesis is written using Quarto with **renv** (Ushey and Wickham 2024) to create a reproducible environment. All materials required to reproduce this document—including data sets, source files, and implementation code—are publicly available in the GitHub repository github.com/DhyeyMavani2003/ccrvam. The repository contains the complete codebase for the CCRVAM methodology, including the test suite that verifies correct implementation, and rigorous documentation for usability spanning 5000+ lines of code.

Contributors are welcome to submit issues or pull requests to the GitHub repository to help improve the methodology and its implementation.

A.1 Chapter 2 Code

The following code was used to create Chapter 2. The motivating example in Chapter 2 demonstrates how correlation measures can vary dramatically under non-linear transformations while copula-based

measures remain invariant. The code generates bivariate normal data and transforms it via gamma and beta distributions to illustrate this phenomenon:

1. Code for the Figure 2.1.

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import beta, expon, norm, gamma, binom
import os

fig_dir = "fig"
os.makedirs(fig_dir, exist_ok=True)

np.random.seed(8990)
n = 10000
mean = [0, 0]
cov = [[1, 0.8], [0.8, 1]]
X = np.random.multivariate_normal(mean, cov, size=n)
X1, X2 = X[:, 0], X[:, 1]

# Transform U_X1 and U_X2 to uniform [0, 1]
# using the CDF of the normal distribution
U_X1 = norm.cdf(X1)
U_X2 = norm.cdf(X2)

# Transform U_X1 and U_X2 into Gamma and Beta distributions
Y1 = gamma.ppf(U_X1, a=3, scale=1/15)
Y2 = beta.ppf(U_X2, a=5, b=3)

# Pearson Correlation Coefficients
rho_X = np.corrcoef(X1, X2)[0, 1]
rho_Y = np.corrcoef(Y1, Y2)[0, 1]
print("Pearson correlation for (X1, X2):", rho_X)
print("Pearson correlation for (Y1, Y2):", rho_Y)

fig, axes = plt.subplots(1, 2, figsize=(6, 4))

# Scatter plot for (X1, X2)
axes[0].scatter(X1, X2, alpha=0.3, s=5)
axes[0].set_title("Scatter plot of (X1, X2)")
axes[0].set_xlabel("X1")
axes[0].set_ylabel("X2")

axes[0].hist(
  X1, bins=50, density=True, alpha=0.9, color='blue',
  orientation='vertical', histtype='step'
)
axes[0].hist(
  X2, bins=50, density=True, alpha=0.9, color='red',
  histtype='step', orientation='horizontal'
```

```
)

# Scatter plot for (Y1, Y2)
axes[1].scatter(Y1, Y2, alpha=0.3, s=5)
axes[1].set_title("Scatter plot of (Y1, Y2)")
axes[1].set_xlabel("Y1")
axes[1].set_ylabel("Y2")

axes[1].hist(
  Y1, bins=50, density=True, alpha=0.9, color='blue',
  orientation='vertical', histtype='step'
)
axes[1].hist(
  Y2, bins=50, density=True, alpha=0.9, color='red',
  histtype='step', orientation='horizontal'
)

plt.tight_layout()
fig_path = os.path.join(fig_dir, "motivating_example.png")
plt.savefig(fig_path, dpi=300, bbox_inches='tight')

plt.close(fig)
```

2. Code for the Figure 2.2.

```
np.random.seed(8990)

# Apply PIT to all variables in order to make them uniform
U_Y1 = gamma.cdf(Y1, a=3, scale=1/15)
U_Y2 = beta.cdf(Y2, a=5, b=3)

# Pearson Correlation Coefficients
rho_U_X = np.corrcoef(U_X1, U_X2)[0, 1]
rho_U_Y = np.corrcoef(U_Y1, U_Y2)[0, 1]
print("Pearson correlation for ($F_1(X_1)$, $F_2(X_2)$):", rho_U_X)
print("Pearson correlation for ($G_1(Y_1)$, $G_2(Y_2)$):", rho_U_Y)

uniform_data = np.vstack([U_X1, U_X2, U_Y1, U_Y2]).T

# Verify the uniformity of transformed data (Should be 0.5 in value)
print("U_X1 mean:", U_X1.mean(), "U_X2 mean:", U_X2.mean())
print("U_Y1 mean:", U_Y1.mean(), "U_Y2 mean:", U_Y2.mean())

fig, axes = plt.subplots(1, 2, figsize=(6, 4))

# Scatter plot for (U_X1, U_X2)
axes[0].scatter(U_X1, U_X2, alpha=0.3, s=5)
axes[0].set_title("Scatter plot of ($F_1(X_1)$, $F_2(X_2)$)")
axes[0].set_xlabel("$F_1(X_1)$")
axes[0].set_ylabel("$F_2(X_2)$")
```

```

# Add marginal histograms
axes[0].hist(
  U_X1, bins=50, density=True, alpha=0.9, color='blue',
  orientation='vertical', histtype='step'
)
axes[0].hist(
  U_X2, bins=50, density=True, alpha=0.9, color='red',
  histtype='step', orientation='horizontal'
)

# Scatter plot for (U_Y1, U_Y2)
axes[1].scatter(U_Y1, U_Y2, alpha=0.3, s=5)
axes[1].set_title("Scatter plot of ($G_1(Y_1)$, $G_2(Y_2)$)")
axes[1].set_xlabel("$G_1(Y_1)$")
axes[1].set_ylabel("$G_2(Y_2)$")

axes[1].hist(
  U_Y1, bins=50, density=True, alpha=0.9, color='blue',
  orientation='vertical', histtype='step'
)
axes[1].hist(
  U_Y2, bins=50, density=True, alpha=0.9, color='red',
  histtype='step', orientation='horizontal'
)

plt.tight_layout()

fig_path = os.path.join(fig_dir, "transformed_motivating_example.png")
plt.savefig(fig_path, dpi=300, bbox_inches='tight')

plt.close(fig)
```

3. Code for the Figure 2.3.

```
np.random.seed(8990)

# Transform (Y1,Y2) back to normal marginals using quantile transformation
F1_Y1 = norm.ppf(gamma.cdf(Y1, a=3, scale=1/15))
F2_Y2 = norm.ppf(beta.cdf(Y2, a=5, b=3))

# Pearson Correlation Coefficients
rho_F_Y = np.corrcoef(F1_Y1, F2_Y2)[0, 1]
print("Pearson correlation for transformed:", rho_F_Y)
print("Pearson correlation between X1 and X2:", rho_X)

fig, axes = plt.subplots(1, 2, figsize=(6, 4))

# Scatter plot for original normal marginals (X1, X2)
axes[0].scatter(X1, X2, alpha=0.3, s=10)
```

```
axes[0].set_title("Scatter plot ($F_1(X_1)$, $F_2(X_2)$)")
axes[0].set_xlabel("$F_1(X_1)$")
axes[0].set_ylabel("$F_2(X_2)$")
axes[0].hist(
  X1, bins=50, density=True, alpha=0.6, color='blue', histtype='step'
)
axes[0].hist(
  X2, bins=50, density=True, alpha=0.6, color='red',
  histtype='step', orientation='horizontal'
)

# Scatter plot for transformed normal marginals (F1_Y1, F2_Y2)
axes[1].scatter(F1_Y1, F2_Y2, alpha=0.3, s=10)
axes[1].set_title(
  "Scatter plot ($F_1^{-1}(G_1(Y_1))$, $F_2^{-1}(G_2(Y_2))$)"
)
axes[1].set_xlabel("$F_1^{-1}(G_1(Y_1))$")
axes[1].set_ylabel("$F_2^{-1}(G_2(Y_2))$")
axes[1].hist(
  F1_Y1, bins=50, density=True, alpha=0.6, color='blue', histtype='step'
)
axes[1].hist(
  F2_Y2, bins=50, density=True, alpha=0.6, color='red',
  histtype='step', orientation='horizontal'
)

plt.tight_layout()
fig_path = os.path.join(
  fig_dir, "quantile_transformed_motivating_example.png"
)
plt.savefig(fig_path, dpi=300, bbox_inches="tight")
plt.close(fig)
```

A.2 Chapter 3 Code

The following code was used to create Chapter 3. The code below creates the visual representation of a checkerboard copula density function (Figure 3.1) mentioned in an example from Wei and Kim (2021) that serves as the foundation for understanding Wei and Kim (2021)'s methodology:

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.colors as colors

def create_copula_density_plot():
    # Create figure
```

```
    fig, ax = plt.subplots(figsize=(3, 3))

    u1_divisions = np.array([0, 2/8, 3/8, 5/8, 6/8, 1])
    u2_divisions = np.array([0, 2/8, 4/8, 1])

    n_rows = len(u1_divisions) - 1
    n_cols = len(u2_divisions) - 1

    # Create meshgrid for the full plot
    du1 = np.diff(u1_divisions)
    du2 = np.diff(u2_divisions)

    density_values = np.array([
        [0, 0, 2],
        [0, 4, 0],
        [4, 0, 0],
        [0, 4, 0],
        [0, 0, 2],
    ])

    # Create colormap for the specific values (0, 2, 4)
    cmap = colors.ListedColormap(['lightblue', 'violet', 'magenta'])
    bounds = [-0.5, 0.5, 2.5, 4.5]
    norm = colors.BoundaryNorm(bounds, cmap.N)

    for i in range(n_rows):
        for j in range(n_cols):
            value = density_values[i, j]
            rect = plt.Rectangle(
                (u2_divisions[j], u1_divisions[i]),
                du2[j], du1[i],
                facecolor=cmap(norm(value)),
                alpha=1.0,
                edgecolor='black',
                linewidth=0.5
            )
            ax.add_patch(rect)

            ax.text(
                u2_divisions[j] + du2[j]/2,
                u1_divisions[i] + du1[i]/2,
                str(int(value)),
                horizontalalignment='center',
                verticalalignment='center',
                fontsize=16,
                color='black'
            )

    ax.set_xlabel('$U_2$', fontsize=14)
    ax.set_ylabel('$U_1$', fontsize=14)
    ax.set_xlim(0, 1)
    ax.set_ylim(0, 1)

```

```
    for u in u1_divisions:
        ax.axhline(y=u, color='black', linestyle=':', linewidth=1)
    for u in u2_divisions:
        ax.axvline(x=u, color='black', linestyle=':', linewidth=1)

    sm = plt.cm.ScalarMappable(cmap=cmap, norm=norm)
    sm.set_array([])
    cbar = plt.colorbar(sm, ax=ax, ticks=[0, 2, 4])
    cbar.set_label('')

    fig.text(
      0.5, 0.05, "Copula density $c^+(u_1, u_2)$",
      ha='center', fontsize=14
    )

    plt.tight_layout(rect=[0, 0.07, 1, 1])

    return fig

# Generate the plot
fig = create_copula_density_plot()

plt.savefig('fig/copula_density_plot.png', dpi=300, bbox_inches='tight')
plt.close(fig)
```