Indian Statistical Institute, Kolkata

Project for 'Statistical Methods -II'

Bachelors of Statistics (Hons.), 2023-26

Cross Correlation

Adeesh Ajay Devasthale	BS2303
Atmadeep Sengupta	BS2320
Nishant Lamboria	BS2334
Ronak Gupta	BS2339
Srijan Bhowmick	BS2352

Advisor/Instructor

DR. ARNAB CHAKRABORTY

Associate Professor, Applied Statistics Unit

Sunday 12th May, 2024

Contents

1 Abstract

This project involves construction of two completely deterministic random variables which mimic a pair of independent random variables in their joint distribution using Nelsen's theorem[1]. We further generate samples from these joint random variables and use them to test for independence using the Chatterjee Correlation Coefficient[2], which claims to detect the level of dependence in a pair of random variables.

We aim to explore whether the Chatterjee Correlation Coefficient is able to detect the dependence in the so constructed pair of random variables.

Although the two theorems have big theories around them, we provide below what is necessary for our purpose.

2 Nelsen's Theorem

Nelsen's theorem states that given two independent random variables (X, Y) both following Unif$(0, 1)$, and $\epsilon > 0$, there exist random variables (U, V) such that they have the same marginals as (X, Y) and the joint distributions at every point are ϵ-close, that is,

$$\forall (p, q) \in [0, 1]^2, \quad |F_{UV}(p, q) - F_{XY}(p, q)| < \epsilon$$

2.1 Intuition

The soul of the proof lies in the following idea:

The total probability mass of (U, V) where $U \sim$ Unif$(0, 1)$ and $V = U$ lies on the $y = x$ line uniformly. To mimic the independent bivariate Unif$(0, 1)$, we need to spread the mass evenly throughout the square $[0, 1] \times [0, 1]$. So given ϵ, we divide the unit-square into smaller squares each with a small side-length depending on ϵ, and then distribute the whole mass (line) equally in all squares. Here we have to ensure the function so formed remains bijective for $V := f(U)$ to also follow Unif$(0, 1)$.

To further strengthen this intuition, consider this example where we divide the unit square in 9 smaller squares, and re-distribute the mass equally while maintaining bijection:

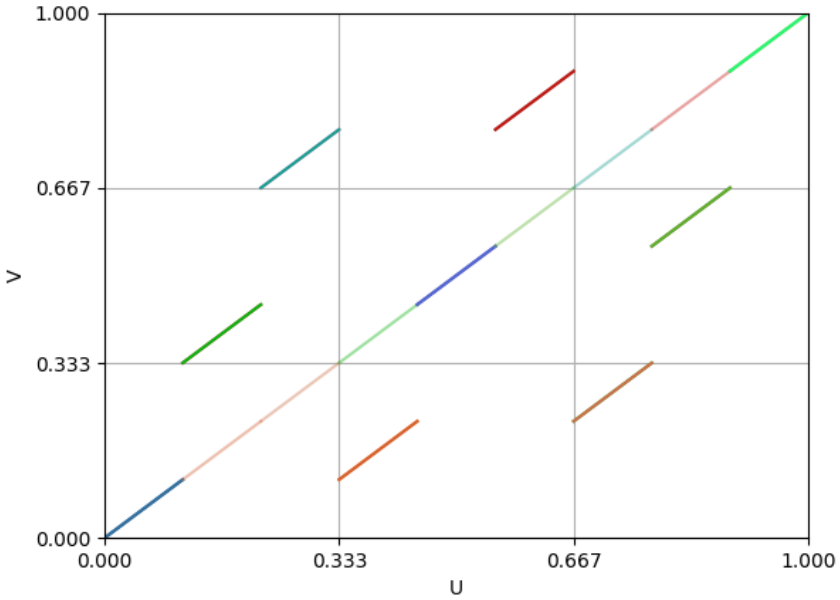

Figure 2.1: Nelsen's Function for $m = 3$

Here, since every piece of the line carries equal mass ($\frac{1}{9}$ in this case), it is easy to verify that the value of the joint CDF at points $\left(\frac{a}{3}, \frac{b}{3}\right)$ where $a, b \in \{0, 1, 2, 3\}$ is $\frac{ab}{9}$, which matches exactly with that of the independent pair. In-between these lattice points, the value of the joint CDF is bounded above and below by that of the lattice points themselves, hence bounding the variation. Therefore, for a given ϵ, we can find a large enough m, divide the unit-square in m^2 smaller squares and ensure ϵ-closeness.

The original proof is due to *Kimeldorf, Sampson*[3] and *Mikusinski*[4]. Here we present a different proof bypassing the notion of Copulas.

2.2 Proof

Let $m > \frac{3}{\epsilon}$ be a natural number, and let $n = m^2$. Let $S = \{1, 2, \ldots, n\}$, and $\pi : S \to S$ such that

$$m(k-1) + j \mapsto m(j-1) + k \text{ where } j, k \in \{1, 2, \ldots m\}$$

Note that π is a bijection since the representation above is unique for every element in S. Moreover, it is easy to see that $\pi = \pi^{-1}$ as $\pi(\pi(a)) = a$.

Let $I_k = (\frac{k-1}{n}, \frac{k}{n}]$, define $f : [0, 1] \to [0, 1]$ such that

$$f(x) = \begin{cases} 0 & x = 0 \\ x + \frac{\pi(k)-k}{n} & x \in I_k, \end{cases}$$

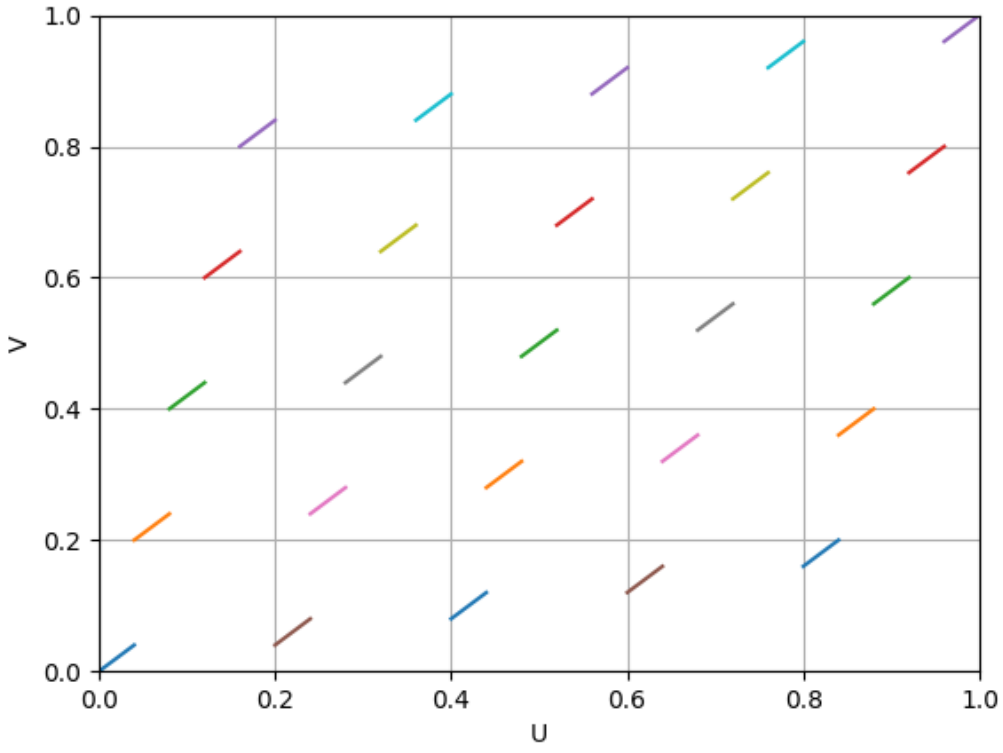

Figure 2.2: Nelsen's Function for $m = 5$

Note that, f maps I_k to $I_{\pi(k)}$ linearly with slope +1 and hence, bijectively, and since π itself is a bijection, we can conclude that f is a bijection. Another notable property of f is that $f = f^{-1}$ as $f(f(I_k)) = f(I_{\pi(k)}) = I_{\pi(\pi(k))} = I_k$

Claim: If $U \sim \text{Unif}(0,1)$ and $V = f(U)$, then $V \sim \text{Unif}(0,1)$.

Proof: Fix $a \in [0,1]$, say $a \in I_k$, then

$$P(V \le a) = P(f(U) \le a) = P(U \in f^{-1}([0,a])) = P(U \in f([0,a]))$$

$$= P\left(U \in f\left(\{0\} \cup \left(\bigcup_{j=1}^{k-1} I_j\right) \cup \left(\frac{k-1}{n}, a\right]\right)\right)$$

$$= P\left(U \in \{0\} \cup \left(\bigcup_{j=1}^{k-1} I_{\pi(j)}\right) \cup \left(\frac{\pi(k)-1}{n}, f(a)\right]\right)$$

$$= \frac{k-1}{n} + f(a) - \frac{\pi(k)-1}{n} = \frac{k-1}{n} + a + \frac{\pi(k)-k}{n} - \frac{\pi(k)-1}{n} = a$$

$$P(V \le a) = a, \ \forall a \in [0,1] \implies V \sim \text{Unif}(0,1). \text{ Hence proved.}$$

Claim: $F_{UV}\left(\frac{a}{m}, \frac{b}{m}\right) = \frac{ab}{n}$ for all $a, b \in \{1, 2, \dots m\}$.

Proof: Note that,

$$F_{UV}\left(\frac{a}{m}, \frac{b}{m}\right) = P\left(U \le \frac{a}{m}, V \le \frac{b}{m}\right) = P\left(U \in \left[0, \frac{a}{m}\right], f(U) \in \left[0, \frac{b}{m}\right]\right)$$

$$= P\left(U \in \left[0, \frac{a}{m}\right] \cap f^{-1}\left(\left[0, \frac{b}{m}\right]\right)\right) = P\left(U \in \left[0, \frac{a}{m}\right] \cap f\left(\left[0, \frac{b}{m}\right]\right)\right)$$

$$= P\left(U \in \left(\bigcup_{j=1}^{ma} I_j\right) \cap \left(\bigcup_{k=1}^{mb} I_{\pi(k)}\right)\right)$$

So, we need the number of intervals from $\left(\bigcup_{k=1}^{mb} I_{\pi(k)}\right)$ that lie in $\left(\bigcup_{j=1}^{ma} I_j\right)$,

$$= \#\left\{k \in \{1, \dots, mb\} : I_{\pi(k)} \subset \left(\bigcup_{j=1}^{ma} I_j\right)\right\} = \#\{k \in \{1, \dots, mb\} : \pi(k) \le ma\}$$

but, $\{1, \dots, mb\} = \{m(l-1) + r : l \in \{1, \dots, b\}, r \in \{1, \dots, m\}\}$

$$\implies \{\pi(k) : k \in \{1, \dots, mb\}\} = \{m(r-1) + l : l \in \{1, \dots, b\}, r \in \{1, \dots, m\}\}$$

$$\therefore \ \#\{k \in \{1, \dots, mb\} : \pi(k) \le ma\} = \#\{m(r-1) + l : l \in \{1, \dots, b\}, r \in \{1, \dots, a\}\} = ab$$

Since $P(U \in I_k) = 1/n$ and all the intervals are disjoint, we get $F_{UV}\left(\frac{a}{m}, \frac{b}{m}\right) = \frac{ab}{n}$. Hence proved.

Now, for any $(p,q) \in [0,1]^2$, take $i = \lfloor pm \rfloor$ and $j = \lfloor qm \rfloor$. Now, as F_{UV} is increasing in both arguments,

$$\implies F_{UV}\left(\frac{i}{m}, \frac{j}{m}\right) \le F_{UV}(p,q) \le F_{UV}\left(\frac{i+1}{m}, \frac{j+1}{m}\right)$$

$$\implies \frac{ij}{n} \le F_{UV}(p,q) \le \frac{(i+1)(j+1)}{n}$$

Given (X, Y) are two independent random variables both following $\text{Unif}(0,1)$, we get $F_{XY}(p,q) = pq$. So,

$$\frac{ij}{n} - pq \le F_{UV}(p,q) - F_{XY}(p,q) \le \frac{(i+1)(j+1)}{n} - pq$$

$$\implies |F_{UV}(p,q) - F_{XY}(p,q)| \leq \max\left\{\left|\frac{ij}{n} - pq\right|, \left|\frac{(i+1)(j+1)}{n} - pq\right|\right\}$$

but,

$$\left|\frac{ij}{n} - pq\right| \leq \left|\frac{ij}{n} - \frac{(i+1)(j+1)}{n}\right| = \frac{(i+j+1)}{n} = \left|\frac{(i+1)(j+1)}{n} - \frac{ij}{n}\right| \geq \left|\frac{(i+1)(j+1)}{n} - pq\right|$$

therefore,

$$\boxed{|F_{UV}(p,q) - F_{XY}(p,q)| \leq \frac{(i+j+1)}{n} \leq \frac{3m}{n} = \frac{3}{m} < \epsilon}$$

Hence proved.

2.3 Remarks

It is worth mentioning that in construction of f above, we are not restricted to keep slope of every piece as $+1$. As long as we can ensure uniformity and bijection, we are free to modify f. However, these restrictions limit us to choosing the slopes of every piece from only $\{+1, -1\}$. This gives us a bigger class of functions to choose from. Here is an example of such a function:

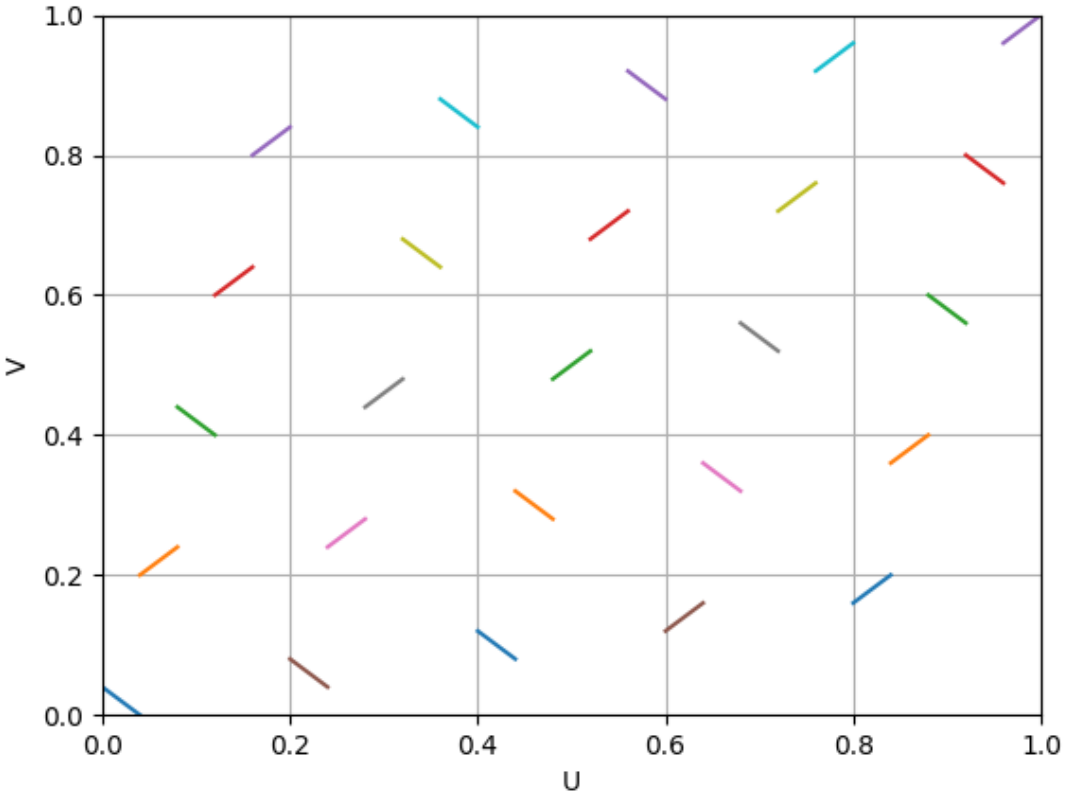

Figure 2.3: Nelsen's Function for $m = 5$ (with randomized slopes)

3 Chatterjee Correlation Coefficient

The *Chatterjee Correlation Coefficient*, developed and published by *Sourav Chatterjee* in 2021, aims to be a reliable measure of the strength of the relationship between 2 random variables unlike the classical coefficients such as Pearson, Spearman, etc while having a relatively simple formula.

3.1 Definition

Let (X, Y) be a pair of random variables, where Y is not a constant. Let $(X_1, Y_1), \ldots, (X_n, Y_n)$ be i.i.d. pairs with the same law as (X, Y), where $n \geq 2$. Rearrange the data as $\left(X_{(1)}, Y_{(1)}\right), \ldots, \left(X_{(n)}, Y_{(n)}\right)$ such that $X_{(1)} \leq \cdots \leq X_{(n)}$ If there are ties , X_i 's , then choose an increasing rearrangement as above by breaking ties uniformly at random. Let r_i be the rank of $Y_{(i)}$, that is, the number of j such that $Y_{(j)} \leq Y_{(i)}$. Let l_i be the number of j such that $Y_{(j)} \geq Y_{(i)}$. Then Chatterjee correlation coefficient is given by

$$\xi_n(X, Y) := 1 - \frac{n \sum_{i=1}^{n-1} |r_{i+1} - r_i|}{2 \sum_{i=1}^{n} l_i (n - l_i)}.$$

Chatterjee Correlation coefficient has a simpler formula if the X_i 's and the Y_i 's have no ties . When there are no ties among the Y_i 's , $l_1, \ldots, l_n$ is just a permutation of $1, \ldots, n$, and so the denominator in the above expression is just $n\left(n^2 - 1\right)/3$. Hence , Chatterjee Correlation Coefficient has the formula

$$\xi_n(X, Y) := 1 - \frac{3 \sum_{i=1}^{n-1} |r_{i+1} - r_i|}{n^2 - 1}.$$

3.2 Some properties

- ξ_n is not symmetric in X and Y. It helps to understand if Y is a function X, and not just if one of the variables is a function of the other. If we want to understand whether X is a function of Y, we should use $\xi_n(Y, X)$ instead of $\xi_n(X, Y)$.
- In theory, Chatterjee correlation coefficient is 0 if and only if X and Y are independent, and 1 if and only if at least one of X and Y is a measurable function of the other.
- The coefficient $\xi_n(X, Y)$ remains unchanged if we apply strictly increasing transformations to X and Y, because it is based on ranks. For the same reason, it can be computed in time $O(n \log n)$.
- If the X_i 's have ties, then $\xi_n(X, Y)$ is a randomized estimate of $\xi(X, Y)$, because of the randomness coming from the breaking of ties.

3.3 Test for Independence

The Chatterjee Correlation Coefficient has an asymptotic theory under the hypothesis of independence. An important theorem for practical purposes is presented without proof below. It gives the asymptotic distribution of $\sqrt{n}\xi_n$ under the hypothesis of independence and the assumption that Y is continuous.

> Suppose that X and Y are independent and Y is continuous.
> Then $\sqrt{n}\xi_n(X, Y) \to N(0, 2/5)$ in distribution as $n \to \infty$.

3.4 Usage

To test a random sample for independence, we set our null hypothesis, H_0: *X and Y are independent*, with the alternative H_a: *X and Y are not completely independent*. Then we calculate the realized value of $\sqrt{n}\xi_n$ and find the *p-value*, and the bound B for the 95% confidence interval with the help of the above mentioned theorem.

4 Exploration

We used R to do our exploration. The coding part included:

- Constructing U and V as in the proof in Section 2.1 (with randomized slopes).
- Generating random samples from (U, V) and (X, Y).
- Coding the Chatterjee Correlation Coefficient calculator.
- Testing the output for independence.

4.1 Sample Outputs

Here are some sample outputs from our code:

Sample Output 1.1: $\epsilon = 0.01$, $N = 500$

```
# Exploring independent data #

  Characteristic        Value
1            CCC -0.01054804
2             |Z|  0.23586139
3        p-value  0.70920083
4              B  1.23959006


## Exploring dependent data ##
  Characteristic       Value
1            CCC 0.03193213
2             |Z| 0.71402408
3        p-value 0.25890998
4              B 1.23959006
```

Sample Output 1.2: $\epsilon = 0.01$, $N = 500$

```
# Exploring independent data #

  Characteristic        Value
1            CCC -0.01014004
2             |Z|  0.22673820
3        p-value  0.71996575
4              B  1.23959006


## Exploring dependent data ##

  Characteristic        Value
1            CCC 0.096276385
2             |Z| 2.152805417
3        p-value 0.000664349
4              B 1.239590065
```

Sample Output 2.1: $\epsilon = 0.01$, $N = 1000$

```
# Exploring independent data #

  Characteristic       Value
1            CCC 0.01144201
2             |Z| 0.36182817
3        p-value 0.56725384
4              B 1.23959006


## Exploring dependent data ##

  Characteristic         Value
1            CCC 1.384501e-01
2             |Z| 4.378178e+00
3        p-value 4.437201e-12
4              B 1.239590e+00
```

Sample Output 2.2: $\epsilon = 0.01$, $N = 1000$

```
# Exploring independent data #

  Characteristic       Value
1            CCC 0.04652105
2             |Z| 1.47112466
3        p-value 0.02001576
4              B 1.23959006


## Exploring dependent data ##

  Characteristic         Value
1            CCC 1.265551e-01
2             |Z| 4.002024e+00
3        p-value 2.487515e-10
4              B 1.239590e+00
```

Sample Output 3.1: $\epsilon = 0.01$, $N = 5000$

```
# Exploring independent data #

  Characteristic      Value
1            CCC 0.00094836
2             |Z| 0.06705918
3        p-value 0.91555865
4              B 1.23959006


## Exploring dependent data ##

  Characteristic      Value
1            CCC  0.6868494
2             |Z| 48.5675888
3        p-value  0.0000000
4              B  1.2395901
```

Sample Output 3.2: $\epsilon = 0.01$, $N = 5000$

```
# Exploring independent data #

  Characteristic      Value
1            CCC 0.00546072
2             |Z| 0.38613123
3        p-value 0.54151271
4              B 1.23959006


## Exploring dependent data ##

  Characteristic     Value
1            CCC  0.683829
2             |Z| 48.354014
3        p-value  0.000000
4              B  1.239590
```

Sample Output 4.1: $\epsilon = 0.001$, $N = 5000$

```
# Exploring independent data #

  Characteristic     Value
1            CCC 0.0126312
2             |Z| 0.8931608
3        p-value 0.1578878
4              B 1.2395901


## Exploring dependent data ##

  Characteristic        Value
1            CCC 4.592832e-02
2             |Z| 3.247623e+00
3        p-value 2.822303e-07
4              B 1.239590e+00
```

Sample Output 4.2: $\epsilon = 0.001$, $N = 5000$

```
# Exploring independent data #

  Characteristic      Value
1            CCC 0.00729612
2             |Z| 0.51591361
3        p-value 0.41465398
4              B 1.23959006


## Exploring dependent data ##

  Characteristic        Value
1            CCC 4.476012e-02
2             |Z| 3.165019e+00
3        p-value 5.605557e-07
4              B 1.239590e+00
```

Sample Output 5.1: $\epsilon = 0.001$, $N = 10000$

```
# Exploring independent data #

  Characteristic       Value
1            CCC -0.00543606
2             |Z|  0.54360601
3        p-value  0.39005559
4              B  1.23959006


## Exploring dependent data ##

  Characteristic         Value
1            CCC  1.479922e-01
2             |Z|  1.479922e+01
3        p-value 4.311810e-121
4              B  1.239590e+00
```

Sample Output 5.2: $\epsilon = 0.001$, $N = 10000$

```
# Exploring independent data #

  Characteristic       Value
1            CCC -0.00384474
2             |Z|  0.38447400
3        p-value  0.54324930
4              B  1.23959006


## Exploring dependent data ##

  Characteristic         Value
1            CCC  1.400327e-01
2             |Z|  1.400327e+01
3        p-value 1.270524e-108
4              B  1.239590e+00
```

4.2 Observations

We observed that given enough samples, the Chatterjee Correlation Coefficient is able to distinguish between the dependent and independent pairs by a significant margin. However, for lesser number of points, the test is inconclusive for both as the variance is high in the outputs

5 Conclusion

The exploration led to the easy conclusion that the Chatterjee Correlation Coefficient is able to detect the dependence between the random variables constructed through Nelsen's proof given sufficient number of sample points. This is because the Chatterjee Correlation Coefficient works with ranked samples and is thus affected greatly by monotony of the joint random variable.

While at first sight there may seem to be a "contradiction" between the two theorems, it is inherently nonexistent as Nelsen's construction disguises the joint-CDF of the pair, but the monotony is well-maintained in the pieces, which is detected by the CCC.

A Appendix

A.1 Code

```
# Defining control parameters #

epsilon = 0.01
N = 1000      # Number of observations to be generated #


################################################################################
############## Code to generate V given U in view of Nelsen's Theorem ##############
################################################################################

# Defining necessary parameters #
m = ceiling(3/epsilon)
n = m**2
S = 1:n

# Generating the permutation as needed #
P = c()
L = m*(0:(m-1))
for(a in 1:m){
  P = append(P, a + L)
}

# Randomizing slopes of every piece-wise function #
W = sample(c(1,-1), n, rep = T, prob = c(0.5, 0.5))

# Function to convert a vector of U-values to their corresponding V-values #
U_V = function(Vector){
  Output = c()     # Initializing vector to store the output #
  for(u in Vector){
    if(u==0){
      Output = append(Output, 0)     # Handling the only boundary case #
    }
    else{
      k = ceiling(n*u)     # "Piece number" of the value #
      r = k/n - u      # "Error" from the upper boundary of the piece #
      v = (P[k]+(W[k]-1)/2)/n - W[k]*r    # Calculating v #
      Output = append(Output, v) # Appending to the output vector #
    }
  }
```

```
  return(Output)
}


##############################################################################
#################### Code to generate dependent data (U and V) ###############
##############################################################################

dep = function(n){     # n = number of observations needed #
  U = runif(n)         # Generating U ~ Unif(0,1) #

  dD = data.frame(     # Creating a data-frame with U and V as columns #
    U = U,
    V = U_V(U)         #  Generating V given U #
  )

  return(dD)
}

dData = dep(N)    # Storing the data in a variable #


##############################################################################
#################### Code to create independent data (X and Y) ###############
##############################################################################

ind = function(n){       # n = number of observations needed #
  X = runif(n)           # Generating X ~ Unif(0,1) #
  Y = runif(n)           # Generating Y ~ Unif(0,1) #

  iD = data.frame(       # Creating a data-frame with X and Y as columns #
    X = X,
    Y = Y
  )

  return(iD)
}

iData = ind(N)       # Storing the data in a variable #
#iData[,1] = U       # For specific tests #

##############################################################################
############## Code to calculate the Chatterjee Correlation Coefficient ##############
######################### given bivariate data in a data-frame ###############################
##############################################################################

# Function to handle (randomize) ties in the data #
randomize = function(D){     # D = Data #

  L = D[,1]          # First column of the data #

  # The following code assumes the entries in the first column are ordered, #
  # which is ensured naturally by the CCC calculation algorithm #

  i = 1              # Indicator running over the length of L #
  while(i < N){        # Stop condition #

    j = i            # Sub-indicator to check for repetitions #
    while(j < N && L[j+1] == L[i]){
      j = j + 1
    }                # Stops when 'j' is the last index with repetition #

    if(j > i){         # If there are repetitions of an observation #
```

```
      # Random permutation of numbers from 1 to 'no. of repetitions' #
      r = sample(j-i+1)
      # Applying the permutation to the repeated part of the data #
      D[i:j,] = D[i:j,][r,]
    }

    i = j + 1      # Indicating the next unchecked index #
  }
  return(D)
}

# Chatterjee Correlation Coefficient Calculator #
CCC = function(D){       # D = Data #

  # Arranging the data #
  D = D[order(D[,1]),]  # Ordering the first column #
  D = randomize(D)      # Randomizing in case of ties #

  # Calculating ranks of the second column #
  R = rank(D[,2], ties.method = c("max"))      # |{Y: Y <= Y_i}| = r_i #
  L1 = rank(D[,2], ties.method = c("min"))-1   # |{Y: Y < Y_i}| = N - l_i #
  L2 = N - L1                                  # |{Y: Y => Y_i}| = l_i #

  # Calculating the coefficient #
  num = 0   # Initializing variable to store the numerator value #
  den = 0   # Initializing variable to store the denominator value #

  # Adding the required values to the variables #
  for(i in 1:(N-1)){
    num = num + Mod(R[i+1]-R[i])
    den = den + L1[i]*L2[i]
  }
  den = den + L1[N]*L2[N]

  Z = 1 - (N/2)*(num/den)      # Final value of the CCC #
  return(Z)
}


#############################################################################
#################### Code to test for independence of given data ####################
#############################################################################

# Defining a function to summarize essential observations #
explore = function(D){       # D = Data #
  C = CCC(D)                 # CCC of Data #
  Z = sqrt(N)*Mod(C)      # |Observed value| of RV that follows N(0,0.4) #

  summ = data.frame(         # Initializing  a data-frame to store the summary #
    Characteristic = c("CCC",
                       "|Z|",
                       "p-value",
                       "B"),
    Value = c(C,
              Z,
              2*pnorm(Z, sd = sqrt(0.4), lower.tail = FALSE),
              qnorm(0.975, sd = sqrt(0.4))))

    print.data.frame(summ)  # Printing the summary #
}

explore(iData)       # Exploring independent data #
explore(dData)       ## Exploring dependent data ##
#############################################################################
```

References

[1] Roger B. Nelsen, *An Introduction to Copulas*, 2nd ed. [link]

[2] Sourav Chatterjee (2021), *A New Coefficient of Correlation*, Journal of the American Statistical Association, 116:536, 2009-2022, DOI: 10.1080/01621459.2020.1758115 [link]

[3] Monotone Dependence George Kimeldorf, Allan R. Sampson The Annals of Statistics, Vol. 6, No. 4 (Jul., 1978), pp. 895-903 [link]

[4] Shuffles of Min. Piotr Mikusinski; Howard Sherwood; Michael D. Taylor Stochastica (1992) Volume: 13, Issue: 1, page 61-74 ISSN: 0210-7821 [link]