# AsianBarometersIndia\_Functionalized\_Version3

#### Ziyang Zhou, Wendy Olsen

#### 20 August 2024

# Function Encapsulation

Authors: Ziyang Zhou and Prof. Wendy Olsen.  
  
Affiliation: Department of Social Statistics, University of Manchester   
The license is Creative Commons, so please use it and cite this work as Wendy Olsen and Ziyang Zhou, “Ordinal Inputs in a Social Data Science Context”, conference paper at British Society for Population Studies, 2024. Creative Commons, University of Manchester, 2024.

For convenience of use, we divide the whole process of calculating entropy into steps, and then encapsulate each process into a function. The function can easily by used by passing inputs into it and getting the expected output.

Next, we apply this to a series of trials. Some of that work is done in Stata. # Preparing

## Read data

Note: before running this code, set the working directory to the folder that contains your data or change the filepath to your data file’s path.

setwd("C:/data/AsianBaro")

library(haven)

#abindia <- read\_dta("/Users/zhouziyang/Desktop/RA/abindia/W5\_India\_merged\_core\_20220905\_released.dta")

#Note the cleaning of income variables in AB was done, creating revision1 of raw data. But, we are cleaning income in a stata file. So here we use raw original data.

abindia <- read\_dta("C:/data/AsianBaro/raw/AsianBaro2019.dta")

dim(abindia)

## [1] 5318 303

## Select question variables (ordinal)

Select variables from the dataset and show the distinct values that Question 63, 69, 146 have respectively.

Q63. When a mother-in-law and a daughter-in- law come into conflict, even if the mother-in-law is in the wrong, the husband should still persuade his wife to obey his mother.

Q69. If one could have only one child, it is more preferable to have a boy than a girl.

Q146. Women should not be involved in politics as much as men.

q63 = abindia$Q63

q69 = abindia$Q69

q146 = abindia$Q146

q64 = abindia$Q64

q65 = abindia$Q65

### Pre-processing question variables

Please note that for now the Likert\_Encoder function only accepts questions where their unique rank values are 1,2,3,4,7,8,9. Then it gives a matrix with 5 columns (7,8,9 are combined into neutral, which is represented as 3). Thus, before using the function, clean the data to specified format first.

preprocessing <- function(question\_column){

# 1-strongly agree, 2-somewhat agree, is kept the same.

# 4-strongly disagree, is replaced with 5

question\_column <- ifelse(question\_column == 4, 5, question\_column)

# 3-somewhat disagree, is replaced with 4

question\_column <- ifelse(question\_column == 3, 4, question\_column)

# 7, 8, 9 are missing values (neutral), which are replaced with 3

question\_column <- ifelse(question\_column == 7, 3, question\_column)

question\_column <- ifelse(question\_column == 8, 3, question\_column)

question\_column <- ifelse(question\_column == 9, 3, question\_column)

return (question\_column)

}

q63 <- preprocessing(q63)

q69 <- preprocessing(q69)

q146 <- preprocessing(q146)

q64 <- preprocessing(q64)

q65 <- preprocessing(q65)

distinct\_values\_63 <- unique(q63)

distinct\_values\_69 <- unique(q69)

distinct\_values\_146 <- unique(q146)

distinct\_values\_64 <- unique(q64)

distinct\_values\_65 <- unique(q65)

## Select, Pre-processing Sex variable (binary), and Calculate Sex’s Entropy

library(entropy)

sex <- abindia$SE2 # Male is 1, Female is 2

sex <- sex-1 # Transform it so that Male is 0, Female is 1

distinct\_values\_sex <- unique(sex)

df\_sex <- data.frame(

Sex = sex

)

# Create an empty data frame for one-hot encoding

sex\_encoded <- data.frame(matrix(0, nrow = nrow(df\_sex), ncol = length(unique(df\_sex$Sex))))

# Order unique values numerically

ordered\_unique\_values <- sort(unique(sex))

# Set column names based on unique values in 'df'

colnames(sex\_encoded) <- paste("Sex\_", ordered\_unique\_values, sep = "")

# Traverse through the 'df' column and fill in one-hot encoding

for (i in 1:nrow(df\_sex)) {

value <- as.character(df\_sex$Sex[i])

sex\_encoded[i, paste("Sex\_", value, sep = "")] <- 1

}

#sex\_counts <- table(sex)

sex\_counts <- colSums(sex\_encoded)

sex\_entropy <- entropy(sex\_counts)

# Validate the entropy library with mathematical calculations

# Probabilities for each rank

sex\_probs <- sex\_counts / sum(sex\_counts)

# Shannon Entropy Formula

sex\_shannon\_entropy <- -sum(sex\_probs \* log(sex\_probs))

# Print the result

print(paste("The Shannon entropy of Sex calculated via the Shannon formula is: ",as.character(sex\_shannon\_entropy)))

## [1] "The Shannon entropy of Sex calculated via the Shannon formula is: 0.686683427438679"

print(paste("The Shannon entropy of Sex calculated via the entropy package is: ",as.character(sex\_entropy)))

## [1] "The Shannon entropy of Sex calculated via the entropy package is: 0.686683427438679"

## Selecting Education Variable (Cumulative Orinal)

The education variable has 11 distinct values, where 1-10 represents the level of education from lowest to highest, and 99 represents missing value.

edu = abindia$SE5

distinct\_values\_edu <- unique(edu)

print(distinct\_values\_edu)

## <labelled<double>[11]>: se5 Education

## [1] 1 7 2 99 6 9 8 10 3 5 4

##

## Labels:

## value label

## -1 Missing

## 1 No formal education

## 2 Incomplete primary/elementary

## 3 Complete primary/elementary

## 4 Incomplete secondary/high school: technical/vocational type

## 5 Complete secondary/high school: technical/vocational type

## 6 Incomplete secondary/high school

## 7 Complete secondary/high school

## 8 Some university education

## 9 University education completed

## 10 Post-graduate degree

## 11 Other

## 97 Do not understand the question

## 98 Can't choose

## 99 Decline to answer

### Preprocessing

Pre-process the education variable. We want the following re-labelling transformation, where we let Edu\_1 represents: No formal education; Edu\_2 represents: Missing values, Incomplete primary/elementary, Complete primary/elementary; Edu\_3 represents: Incomplete secondary/high school-technical/vocational type, Incomplete secondary/high school; Edu\_4 represents: Complete secondary/high school-technical/vocational type, Complete secondary/high school; Edu\_5 represents: Some university education, University education completed, Post-graduate degree.

# In the new labels:

# all 1s are kept.

#all 99 and then 11's and 2's are replaced with a 1. combined with 2, 9911.

edu <- ifelse(edu == 99, 1, edu)

edu <- ifelse(edu == 11, 1, edu)

edu <- ifelse(edu == 2, 1, edu)

# all 3s are replaced with 2.

edu <- ifelse(edu == 3, 2, edu)

# all 4s and 6s are replaced with 3.

edu <- ifelse(edu == 4, 3, edu)

edu <- ifelse(edu == 6, 3, edu)

# all 5s and 7s are replaced with 4.

edu <- ifelse(edu == 5, 4, edu)

edu <- ifelse(edu == 7, 4, edu)

# all 8s, 9s, and 10s, are replaced with 5.

edu <- ifelse(edu == 8, 5, edu)

edu <- ifelse(edu == 9, 5, edu)

edu <- ifelse(edu == 10, 5, edu)

check<-table(edu)/5318

str(check)

## 'table' num [1:5(1d)] 0.303 0.103 0.117 0.26 0.218

## - attr(\*, "dimnames")=List of 1

## ..$ edu: chr [1:5] "1" "2" "3" "4" ...

#Please keep in mind that without weights, the sample is biased both toward lots of young people and lots who appear here as having preprimary education.

### Scheme 1: Distinct Encoding

df\_edu <- data.frame(

Edu = edu

)

# Create an empty data frame for one-hot encoding

edu\_encoded <- data.frame(matrix(0, nrow = nrow(df\_edu), ncol = length(unique(df\_edu$Edu))))

# Order unique values numerically

ordered\_unique\_values <- sort(unique(edu))

# Set column names based on unique values in 'df\_q63'

colnames(edu\_encoded) <- paste("Edu\_", ordered\_unique\_values, sep = "")

# Traverse through the 'df\_q63' column and fill in one-hot encoding

for (i in 1:nrow(df\_edu)) {

value <- as.character(df\_edu$Edu[i])

edu\_encoded[i, paste("Edu\_", value, sep = "")] <- 1

}

# Display the result

print(edu\_encoded[1:10, , drop = FALSE])

## Edu\_1 Edu\_2 Edu\_3 Edu\_4 Edu\_5

## 1 1 0 0 0 0

## 2 0 0 0 1 0

## 3 1 0 0 0 0

## 4 1 0 0 0 0

## 5 1 0 0 0 0

## 6 0 0 1 0 0

## 7 0 0 1 0 0

## 8 1 0 0 0 0

## 9 0 0 0 0 1

## 10 1 0 0 0 0

# Get the counts for each education levels

edu\_counts = colSums(edu\_encoded)

print("The counts for each education levels are: ")

## [1] "The counts for each education levels are: "

print(edu\_counts)

## Edu\_1 Edu\_2 Edu\_3 Edu\_4 Edu\_5

## 1609 547 624 1381 1157

### Scheme 2: Cumulative Encoding

df\_edu <- data.frame(

Edu = edu

)

# Create an empty data frame for one-hot encoding

edu2\_encoded <- data.frame(matrix(0, nrow = nrow(df\_edu), ncol = length(unique(df\_edu$Edu))))

# Order unique values numerically

ordered\_unique\_values <- sort(unique(edu))

# Set column names based on unique values in 'df\_q63'

colnames(edu2\_encoded) <- paste("Edu\_", ordered\_unique\_values, sep = "")

# Traverse through the 'df\_q63' column and fill in one-hot encoding

for (i in 1:nrow(df\_edu)) {

value <- df\_edu$Edu[i]

while (value>0){

edu2\_encoded[i, paste("Edu\_", as.character(value), sep = "")] <- 1

value = value-1

}

}

# Display the result

print(edu2\_encoded[1:10, , drop = FALSE])

## Edu\_1 Edu\_2 Edu\_3 Edu\_4 Edu\_5

## 1 1 0 0 0 0

## 2 1 1 1 1 0

## 3 1 0 0 0 0

## 4 1 0 0 0 0

## 5 1 0 0 0 0

## 6 1 1 1 0 0

## 7 1 1 1 0 0

## 8 1 0 0 0 0

## 9 1 1 1 1 1

## 10 1 0 0 0 0

# Get the counts for each education levels

edu2\_counts = colSums(edu2\_encoded)

print("The counts for each education levels are: ")

## [1] "The counts for each education levels are: "

print(edu2\_counts)

## Edu\_1 Edu\_2 Edu\_3 Edu\_4 Edu\_5

## 5318 3709 3162 2538 1157

### Calculative Entropy of Education in Scheme1, Scheme2.

library(entropy)

edu\_entropy = entropy(edu\_counts)

# Probabilities for each bin

edu\_probs <- edu\_counts / sum(edu\_counts)

# Shannon Entropy Formula

edu\_shannon\_entropy <- -sum(edu\_probs \* log(edu\_probs))

# Print the result

print(paste("The Shannon entropy of Education in Scheme 1 calculated via the Shannon formula is: ",as.character(edu\_shannon\_entropy)))

## [1] "The Shannon entropy of Education in Scheme 1 calculated via the Shannon formula is: 1.52903364090424"

print(paste("The Shannon entropy of Education in Scheme 1 calculated via the entropy package is: ",as.character(edu\_entropy)))

## [1] "The Shannon entropy of Education in Scheme 1 calculated via the entropy package is: 1.52903364090424"

edu2\_entropy = entropy(edu2\_counts)

# Probabilities for each bin

edu2\_probs <- edu2\_counts / sum(edu2\_counts)

# Shannon Entropy Formula

edu2\_shannon\_entropy <- -sum(edu2\_probs \* log(edu2\_probs))

# Print the result

print(paste("The Shannon entropy of Education in Scheme 2 calculated via the Shannon formula is: ",as.character(edu2\_shannon\_entropy)))

## [1] "The Shannon entropy of Education in Scheme 2 calculated via the Shannon formula is: 1.51114578051434"

print(paste("The Shannon entropy of Education in Scheme 2 calculated via the entropy package is: ",as.character(edu2\_entropy)))

## [1] "The Shannon entropy of Education in Scheme 2 calculated via the entropy package is: 1.51114578051434"

## Selecting Age Variable (Continuous)

The education variable is a continuous variable.

age <- abindia$Se3\_1

print(paste("The minimum age found is", as.character(min(age))))

## [1] "The minimum age found is 18"

print(paste("The maximum age found is", as.character(max(age))))

## [1] "The maximum age found is 98"

### Preprocessing Age Variable

Discretize the age variable by assigning the values into 10 bins using Min-Max method.

num\_bins <- 10

# Compute the bin boundaries using min-max method

bin\_boundaries <- seq(min(age), max(age), length.out = num\_bins + 1)

# Discretize 'age' into bins

age\_bins <- cut(age, breaks = bin\_boundaries, labels = FALSE, left = FALSE)

age\_bins[is.na(age\_bins)] <- 1

# Convert 'age\_bins' into a one-hot encoded matrix

age\_encoded <- matrix(0, nrow = length(age), ncol = num\_bins)

for (i in 1:length(age)) {

age\_encoded[i, age\_bins[i]] <- 1

}

colnames(age\_encoded) <- c("18-26","27-34","35-42","43-50","51-58","59-66","67-74","75-82","83-90","91-98")

# Get the counts for each age bins

age\_counts = colSums(age\_encoded)

print("The counts for each age bins are: ")

## [1] "The counts for each age bins are: "

print(age\_counts)

## 18-26 27-34 35-42 43-50 51-58 59-66 67-74 75-82 83-90 91-98

## 930 1024 1091 933 492 457 236 111 30 14

#y = discretize( age, num\_bins, r=range(age) )

### Calculate Entropy of Age

library(entropy)

age\_entropy = entropy(age\_counts)

# Probabilities for each bin

age\_probs <- age\_counts / sum(age\_counts)

# Shannon Entropy Formula

age\_shannon\_entropy <- -sum(age\_probs \* log(age\_probs))

# Print the result

print(paste("The Shannon entropy of Age calculated via the Shannon formula is: ",as.character(age\_shannon\_entropy)))

## [1] "The Shannon entropy of Age calculated via the Shannon formula is: 1.94741238723464"

print(paste("The Shannon entropy of Age calculated via the entropy package is: ",as.character(age\_entropy)))

## [1] "The Shannon entropy of Age calculated via the entropy package is: 1.94741238723464"

# Functions

## Function for Encoding Likert Scale data table for Individual Variable

### Define function

Likert\_Encoder <- function(question\_column){

df <- data.frame(

Q = question\_column

)

# Create an empty data frame for one-hot encoding

question\_encoded <- data.frame(matrix(0, nrow = nrow(df), ncol = length(unique(df$Q))))

# Order unique values numerically

ordered\_unique\_values <- sort(unique(question\_column))

# Set column names based on unique values in 'df'

colnames(question\_encoded) <- paste("Rank\_", ordered\_unique\_values, sep = "")

# Traverse through the 'df' column and fill in one-hot encoding

for (i in 1:nrow(df)) {

value <- as.character(df$Q[i])

question\_encoded[i, paste("Rank\_", value, sep = "")] <- 1

}

return (question\_encoded)

}

Three examples of using the function and then calculate its entropy:

q63\_encoded <- Likert\_Encoder(q63)

q69\_encoded <- Likert\_Encoder(q69)

q146\_encoded <- Likert\_Encoder(q146)

print("The likert matrix for Question 63 is (first 10 rows): ")

## [1] "The likert matrix for Question 63 is (first 10 rows): "

print(q63\_encoded[1:10, , drop = FALSE])

## Rank\_1 Rank\_2 Rank\_3 Rank\_4 Rank\_5

## 1 0 0 0 1 0

## 2 0 1 0 0 0

## 3 0 0 0 1 0

## 4 1 0 0 0 0

## 5 0 0 0 0 1

## 6 0 0 0 1 0

## 7 0 0 0 1 0

## 8 0 1 0 0 0

## 9 0 0 0 0 1

## 10 0 0 0 0 1

Then sum each column separately to get the counts of occurrences of 1 for each column(each rank).

library(entropy)

q63\_counts = colSums(q63\_encoded)

q63\_entropy = entropy(q63\_counts)

q69\_counts = colSums(q69\_encoded)

q69\_entropy = entropy(q69\_counts)

q146\_counts = colSums(q146\_encoded)

q146\_entropy = entropy(q146\_counts)

#print(paste("The Shannon entropy of Q63 calculated via the entropy package is: ",as.character(q69\_entropy)))

### Validate the one variable entropy result with mathematical calculation

Because the above one-variable entropy results were obtained by the entropy package, we use mathematical formula of Shannon-entropy to validate if those are correct.

# Probabilities for each rank

q63\_probs <- q63\_counts / sum(q63\_counts)

q69\_probs <- q69\_counts / sum(q69\_counts)

q146\_probs <- q146\_counts / sum(q146\_counts)

# Shannon Entropy Formula

q63\_shannon\_entropy <- -sum(q63\_probs \* log(q63\_probs))

q69\_shannon\_entropy <- -sum(q69\_probs \* log(q69\_probs))

q146\_shannon\_entropy <- -sum(q146\_probs \* log(q69\_probs))

# Print the result

print(paste("The Shannon entropy of Q63 calculated via the Shannon formula is: ",as.character(q63\_shannon\_entropy)))

## [1] "The Shannon entropy of Q63 calculated via the Shannon formula is: 1.55181899638059"

print(paste("The Shannon entropy of Q63 calculated via the entropy package is: ",as.character(q63\_entropy)))

## [1] "The Shannon entropy of Q63 calculated via the entropy package is: 1.55181899638059"

print(paste("The Shannon entropy of Q63 calculated via the Shannon formula is: ",as.character(q69\_shannon\_entropy)))

## [1] "The Shannon entropy of Q63 calculated via the Shannon formula is: 1.5489958219084"

print(paste("The Shannon entropy of Q63 calculated via the entropy package is: ",as.character(q69\_entropy)))

## [1] "The Shannon entropy of Q63 calculated via the entropy package is: 1.5489958219084"

print(paste("The Shannon entropy of Q63 calculated via the Shannon formula is: ",as.character(q146\_shannon\_entropy)))

## [1] "The Shannon entropy of Q63 calculated via the Shannon formula is: 1.57413606580525"

print(paste("The Shannon entropy of Q63 calculated via the entropy package is: ",as.character(q146\_entropy)))

## [1] "The Shannon entropy of Q63 calculated via the entropy package is: 1.56384334764931"

## Function for Generating Likert Scale counts for Two Joint Variables

Likert\_Counter2 <- function(question\_encoded1, question\_encoded2){

counts <- matrix(0, nrow = 5, ncol = 5)

for (i in 1:nrow(question\_encoded1)){

index\_1 <- which(question\_encoded1[i, ] == 1)

index\_2 <- which(question\_encoded2[i, ] == 1)

#row\_index <- as.numeric(gsub("q69\_", "", index\_2))

#col\_index <- as.numeric(gsub("q63\_", "", index\_1))

row\_index <- as.numeric(index\_2)

col\_index <- as.numeric(index\_1)

# Increment the value at the specified position

counts[row\_index, col\_index] <- counts[row\_index, col\_index] + 1

#print(paste(index\_1,index\_2)) # Check if the indexs are correct from the two tables

}

colnames(counts) <- paste("RankA\_", 1:5, sep = '')

rownames(counts) <- paste("RankB\_", 1:5, sep = '')

return(counts)

}

### Stage 1

Before using the two-variable function, first you need to encode each individual variable into likert matrix. Example:

q63\_encoded = Likert\_Encoder(q63)

q69\_encoded = Likert\_Encoder(q69)

### Stage 2

After encoding the variables into data tables (matrices), pass the encoded matrices the as inputs to the counter function, then the counter function will return the counts matrix for each joint rank, in a format that is required by the entropy package.   
Afterwards, pass the counts matrix as input into the entropy package’s given function to get the entropy value.

q63\_q69\_counts = Likert\_Counter2(q63\_encoded,q69\_encoded)

library(entropy)

q63\_q69\_entropy = entropy(q63\_q69\_counts)

### Validate the two variable entropy result with mathematical calculation

q63\_q69\_probs <- q63\_q69\_counts / sum(q63\_q69\_counts)

q63\_q69\_shannon\_entropy <- -sum(q63\_q69\_probs \* log(q63\_q69\_probs))

print(paste("The Shannon entropy of Q63\*Q69 calculated via the Shannon formula is: ",as.character(q63\_q69\_shannon\_entropy)))

## [1] "The Shannon entropy of Q63\*Q69 calculated via the Shannon formula is: 2.93951936616594"

print(paste("The Shannon entropy of Q63\*Q69 calculated via the entropy package is: ",as.character(q63\_q69\_entropy)))

## [1] "The Shannon entropy of Q63\*Q69 calculated via the entropy package is: 2.93951936616594"

## Function for Generating Likert Scale counts for Three Joint Variables

Likert\_Counter3 <- function(question\_encoded1, question\_encoded2, question\_encoded3){

# Create a 3 dimensional matrix first

dim1 <- 5

dim2 <- 5

dim3 <- 5

# Initialize values for the matrix

counts <- array(0, dim = c(dim1, dim2, dim3))

for (i in 1:nrow(q63\_encoded)){

index\_1 <- which(q63\_encoded[i, ] == 1)

index\_2 <- which(q69\_encoded[i, ] == 1)

index\_3 <- which(q146\_encoded[i, ] == 1)

#row\_index <- as.numeric(gsub("q69\_", "", index\_2))

#col\_index <- as.numeric(gsub("q63\_", "", index\_1))

x\_index <- as.numeric(index\_2)

y\_index <- as.numeric(index\_1)

z\_index <- as.numeric(index\_3)

# Increment the value at the specified position

counts[x\_index, y\_index, z\_index] <- counts[x\_index, y\_index, z\_index] + 1

#print(paste(index\_1,index\_2, index\_3)) # Check if the indexs are correct from the two tables

}

colnames(counts) <- paste("q63\_", 1:5, sep = '')

rownames(counts) <- paste("q69\_", 1:5, sep = '')

dim3\_names <- paste("q146\_", 1:5, sep = '')

return(counts)

}

### Stage 1

Before using the three-variable function, first you need to encode each individual variable into likert matrix. Example:

q63\_encoded = Likert\_Encoder(q63)

q69\_encoded = Likert\_Encoder(q69)

q146\_encoded = Likert\_Encoder(q146)

### Stage 2

After encoding the variables into data tables (matrices), pass the encoded matrices the as inputs to the counter function, then the counter function will return the counts matrix for each joint rank, in a format that is required by the entropy package.   
Afterwards, pass the counts matrix as input into the entropy package’s given function to get the entropy value.

q63\_q69\_q146\_counts = Likert\_Counter3(q63\_encoded,q69\_encoded,q146\_encoded)

q63\_q69\_q146\_entropy = entropy(q63\_q69\_q146\_counts)

### Validate the three variable entropy result with mathematical calculation

q63\_q69\_q146\_probs <- q63\_q69\_q146\_counts / sum(q63\_q69\_q146\_counts)

q63\_q69\_q146\_shannon\_entropy <- -sum(q63\_q69\_q146\_probs \* log(q63\_q69\_q146\_probs))

print(paste("The Shannon entropy of Q63\*Q69\*Q146 calculated via the Shannon formula is: ",as.character(q63\_q69\_q146\_shannon\_entropy)))

## [1] "The Shannon entropy of Q63\*Q69\*Q146 calculated via the Shannon formula is: 4.37365254768865"

print(paste("The Shannon entropy of Q63\*Q69\*Q146 calculated via the entropy package is: ",as.character(q63\_q69\_q146\_entropy)))

## [1] "The Shannon entropy of Q63\*Q69\*Q146 calculated via the entropy package is: 4.37365254768865"

## Function for Generating Likert Scale counts for Four Joint Variables

Likert\_Counter4 <- function(question\_encoded1, question\_encoded2, question\_encoded3, question\_encoded4){

# Create a 4 dimensional matrix first

dim1 <- 5

dim2 <- 5

dim3 <- 5

dim4 <- 5

# Initialize values for the matrix

counts <- array(0, dim = c(dim1, dim2, dim3, dim4))

for (i in 1:nrow(q63\_encoded)){

index\_1 <- which(question\_encoded1[i, ] == 1)

index\_2 <- which(question\_encoded2[i, ] == 1)

index\_3 <- which(question\_encoded3[i, ] == 1)

index\_4 <- which(question\_encoded4[i, ] == 1)

x\_index <- as.numeric(index\_2)

y\_index <- as.numeric(index\_1)

z\_index <- as.numeric(index\_3)

h\_index <- as.numeric(index\_4)

# Increment the value at the specified position

counts[x\_index, y\_index, z\_index,h\_index] <- counts[x\_index, y\_index, z\_index,h\_index] + 1

#print(paste(index\_1,index\_2, index\_3, index\_4)) # Check if the indexs are correct from the two tables

}

colnames(counts) <- paste("RankA\_", 1:5, sep = '')

rownames(counts) <- paste("RankB\_", 1:5, sep = '')

return(counts)

}

### Stage 1

Before using the four-variable function, first you need to encode each individual variable into likert matrix. Example:

q63\_encoded = Likert\_Encoder(q63)

q69\_encoded = Likert\_Encoder(q69)

q146\_encoded = Likert\_Encoder(q146)

q64\_encoded = Likert\_Encoder(q64)

### Stage 2

After encoding the variables into data tables (matrices), pass the encoded matrices the as inputs to the counter function, then the counter function will return the counts matrix for each joint rank, in a format that is required by the entropy package.   
Afterwards, pass the counts matrix as input into the entropy package’s given function to get the entropy value.

library(entropy)

q63\_q69\_q146\_q64\_counts = Likert\_Counter4(q63\_encoded,q69\_encoded,q146\_encoded,q64\_encoded)

q63\_q69\_q146\_q64\_entropy = entropy(q63\_q69\_q146\_q64\_counts)

### Validate the four variable entropy result with mathematical calculation

q63\_q69\_q146\_q64\_probs <- q63\_q69\_q146\_q64\_counts / sum(q63\_q69\_q146\_q64\_counts)

q63\_q69\_q146\_q64\_shannon\_entropy <- -sum(q63\_q69\_q146\_q64\_probs \* log(q63\_q69\_q146\_q64\_probs))

print(paste("The Shannon entropy of Q63\*Q69\*Q146\*Q64 calculated via the Shannon formula is: ",as.character(q63\_q69\_q146\_q64\_shannon\_entropy)))

## [1] "The Shannon entropy of Q63\*Q69\*Q146\*Q64 calculated via the Shannon formula is: NaN"

print(paste("The Shannon entropy of Q63\*Q69\*Q146\*Q64 calculated via the entropy package is: ",as.character(q63\_q69\_q146\_q64\_entropy)))

## [1] "The Shannon entropy of Q63\*Q69\*Q146\*Q64 calculated via the entropy package is: 5.51626270832119"

For 4 variable and above, The shannon entropy cannot be calculated in R by hand, this may because some probability is so small that its log value approaches infinity. However, it can still be calculated by the entropy package.

## Function for Generating Likert Scale counts for Five Joint Variables

Likert\_Counter5 <- function(question\_encoded1, question\_encoded2, question\_encoded3, question\_encoded4, question\_encoded5){

# Create a 5 dimensional matrix first

dim1 <- 5

dim2 <- 5

dim3 <- 5

dim4 <- 5

dim5 <- 5

# Initialize values for the matrix

counts <- array(0, dim = c(dim1, dim2, dim3, dim4, dim5))

for (i in 1:nrow(q63\_encoded)){

index\_1 <- which(question\_encoded1[i, ] == 1)

index\_2 <- which(question\_encoded2[i, ] == 1)

index\_3 <- which(question\_encoded3[i, ] == 1)

index\_4 <- which(question\_encoded4[i, ] == 1)

index\_5 <- which(question\_encoded5[i, ] == 1)

x\_index <- as.numeric(index\_2)

y\_index <- as.numeric(index\_1)

z\_index <- as.numeric(index\_3)

h\_index <- as.numeric(index\_4)

i\_index <- as.numeric(index\_5)

# Increment the value at the specified position

counts[x\_index, y\_index, z\_index,h\_index,i\_index] <- counts[x\_index, y\_index, z\_index,h\_index,i\_index] + 1

#print(paste(index\_1,index\_2, index\_3, index\_4)) # Check if the indexs are correct from the two tables

}

colnames(counts) <- paste("RankA\_", 1:5, sep = '')

rownames(counts) <- paste("RankB\_", 1:5, sep = '')

return(counts)

}

### Stage 1

Before using the five-variable function, first you need to encode each individual variable into likert matrix. Example:

q63\_encoded = Likert\_Encoder(q63)

q69\_encoded = Likert\_Encoder(q69)

q146\_encoded = Likert\_Encoder(q146)

q64\_encoded = Likert\_Encoder(q64)

q65\_encoded = Likert\_Encoder(q65)

### Stage 2

After encoding the variables into data tables (matrices), pass the encoded matrices the as inputs to the counter function, then the counter function will return the counts matrix for each joint rank, in a format that is required by the entropy package.   
Afterwards, pass the counts matrix as input into the entropy package’s given function to get the entropy value.

library(entropy)

q63\_q69\_q146\_q64\_q65\_counts = Likert\_Counter5(q63\_encoded,q69\_encoded,q146\_encoded,q64\_encoded,q65\_encoded)

q63\_q69\_q146\_q64\_q65\_entropy = entropy(q63\_q69\_q146\_q64\_q65\_counts)

print(paste("The Shannon entropy of Q63\*Q69\*Q146\*Q64\*Q65 calculated via the entropy package is: ",as.character(q63\_q69\_q146\_q64\_q65\_entropy)))

## [1] "The Shannon entropy of Q63\*Q69\*Q146\*Q64\*Q65 calculated via the entropy package is: 6.34150987096822"

## Function to calculate the entropy of any 2,3,4,5 variable, however the input has to be matrices that are one-hot encoded.

### For example, to calculate the joint entropy of Question 63, Question 69, Education (Scheme 1 and 2) Sex and Age, get the encoded matrix of each:

### Step 1: Preprocessing variables into matrices

# Question 63

q63\_encoded <- Likert\_Encoder(q63)

# Question 69

q69\_encoded <- Likert\_Encoder(q69)

# Edu in Scheme 1 - Distinct

df\_edu <- data.frame(

Edu = edu

)

# Create an empty data frame for one-hot encoding

edu\_encoded <- data.frame(matrix(0, nrow = nrow(df\_edu), ncol = length(unique(df\_edu$Edu))))

# Order unique values numerically

ordered\_unique\_values <- sort(unique(edu))

# Set column names based on unique values in 'df\_q63'

colnames(edu\_encoded) <- paste("Edu\_", ordered\_unique\_values, sep = "")

# Traverse through the 'df\_q63' column and fill in one-hot encoding

for (i in 1:nrow(df\_edu)) {

value <- as.character(df\_edu$Edu[i])

edu\_encoded[i, paste("Edu\_", value, sep = "")] <- 1

}

# Edu in Scheme 2 - Cumulative

df\_edu <- data.frame(

Edu = edu

)

# Create an empty data frame for one-hot encoding

edu2\_encoded <- data.frame(matrix(0, nrow = nrow(df\_edu), ncol = length(unique(df\_edu$Edu))))

# Order unique values numerically

ordered\_unique\_values <- sort(unique(edu))

# Set column names based on unique values in 'df\_q63'

colnames(edu2\_encoded) <- paste("Edu\_", ordered\_unique\_values, sep = "")

# Traverse through the 'df\_q63' column and fill in one-hot encoding

for (i in 1:nrow(df\_edu)) {

value <- df\_edu$Edu[i]

while (value>0){

edu2\_encoded[i, paste("Edu\_", as.character(value), sep = "")] <- 1

value = value-1

}

}

# Sex

df\_sex <- data.frame(

Sex = sex

)

sex\_encoded <- data.frame(matrix(0, nrow = nrow(df\_sex), ncol = length(unique(df\_sex$Sex))))

ordered\_unique\_values <- sort(unique(sex))

colnames(sex\_encoded) <- paste("Sex\_", ordered\_unique\_values, sep = "")

for (i in 1:nrow(df\_sex)) {

value <- as.character(df\_sex$Sex[i])

sex\_encoded[i, paste("Sex\_", value, sep = "")] <- 1

}

# Age

num\_bins <- 10

# Compute the bin boundaries using min-max method

bin\_boundaries <- seq(min(age), max(age), length.out = num\_bins + 1)

# Discretize 'age' into bins

age\_bins <- cut(age, breaks = bin\_boundaries, labels = FALSE, left = FALSE)

age\_bins[is.na(age\_bins)] <- 1

# Convert 'age\_bins' into a one-hot encoded matrix

age\_encoded <- matrix(0, nrow = length(age), ncol = num\_bins)

for (i in 1:length(age)) {

age\_encoded[i, age\_bins[i]] <- 1

}

colnames(age\_encoded) <- c("18-26","27-34","35-42","43-50","51-58","59-66","67-74","75-82","83-90","91-98")

### Step 2: Define 2-variable function

Two\_Var\_Entropy <- function(m1,m2){

# Create a 2 dimensional matrix first

dim1 <- ncol(m1)

dim2 <- ncol(m2)

# Initialize values for the matrix

counts <- array(0, dim = c(dim1, dim2))

for (i in 1:nrow(m1)){

index\_1 <- which(m1[i, ] == 1)

index\_2 <- which(m2[i, ] == 1)

x\_index <- as.numeric(index\_1)

y\_index <- as.numeric(index\_2)

# Increment the value at the specified position

counts[x\_index, y\_index] <- counts[x\_index, y\_index] + 1

}

library(entropy)

strimmer\_entropy <- entropy(counts)

print(paste("The Shannon entropy calculated via the Strimmer Package is: ",as.character(strimmer\_entropy)))

probs <- counts / sum(counts)

shannon\_entropy <- -sum(probs \* log(probs))

print(paste("The Shannon entropy calculated via the Shannon formula is: ",as.character(shannon\_entropy)))

return (entropy(counts))

}

### Step 2: Define 3-variable function

Three\_Var\_Entropy <- function(m1,m2,m3){

# Create a 3 dimensional matrix first

dim1 <- ncol(m1)

dim2 <- ncol(m2)

dim3 <- ncol(m3)

# Initialize values for the matrix

counts <- array(0, dim = c(dim1, dim2, dim3))

for (i in 1:nrow(m1)){

index\_1 <- which(m1[i, ] == 1)

index\_2 <- which(m2[i, ] == 1)

index\_3 <- which(m3[i, ] == 1)

x\_index <- as.numeric(index\_1)

y\_index <- as.numeric(index\_2)

z\_index <- as.numeric(index\_3)

# Increment the value at the specified position

counts[x\_index, y\_index, z\_index] <- counts[x\_index, y\_index, z\_index] + 1

}

library(entropy)

strimmer\_entropy <- entropy(counts)

print(paste("The Shannon entropy calculated via the Strimmer Package is: ",as.character(strimmer\_entropy)))

probs <- counts / sum(counts)

shannon\_entropy <- -sum(probs \* log(probs))

print(paste("The Shannon entropy calculated via the Shannon formula is: ",as.character(shannon\_entropy)))

return (entropy(counts))

}

### Step 2: Define 4-variable function

Four\_Var\_Entropy <- function(m1,m2,m3,m4){

# Create a 4 dimensional matrix first

dim1 <- ncol(m1)

dim2 <- ncol(m2)

dim3 <- ncol(m3)

dim4 <- ncol(m4)

# Initialize values for the matrix

counts <- array(0, dim = c(dim1, dim2, dim3, dim4))

for (i in 1:nrow(m1)){

index\_1 <- which(m1[i, ] == 1)

index\_2 <- which(m2[i, ] == 1)

index\_3 <- which(m3[i, ] == 1)

index\_4 <- which(m4[i, ] == 1)

x\_index <- as.numeric(index\_1)

y\_index <- as.numeric(index\_2)

z\_index <- as.numeric(index\_3)

h\_index <- as.numeric(index\_4)

# Increment the value at the specified position

counts[x\_index, y\_index, z\_index, h\_index] <- counts[x\_index, y\_index, z\_index, h\_index] + 1

}

library(entropy)

strimmer\_entropy <- entropy(counts)

print(paste("The Shannon entropy calculated via the Strimmer Package is: ",as.character(strimmer\_entropy)))

probs <- counts / sum(counts)

shannon\_entropy <- -sum(probs \* log(probs))

print(paste("The Shannon entropy calculated via the Shannon formula is: ",as.character(shannon\_entropy)))

return (entropy(counts))

}

### Step 2: Define 5-variable function

Five\_Var\_Entropy <- function(m1,m2,m3,m4,m5){

# Create a 5 dimensional matrix first

dim1 <- ncol(m1)

dim2 <- ncol(m2)

dim3 <- ncol(m3)

dim4 <- ncol(m4)

dim5 <- ncol(m5)

# Initialize values for the matrix

counts <- array(0, dim = c(dim1, dim2, dim3, dim4, dim5))

for (i in 1:nrow(m1)){

index\_1 <- which(m1[i, ] == 1)

index\_2 <- which(m2[i, ] == 1)

index\_3 <- which(m3[i, ] == 1)

index\_4 <- which(m4[i, ] == 1)

index\_5 <- which(m5[i, ] == 1)

x\_index <- as.numeric(index\_1)

y\_index <- as.numeric(index\_2)

z\_index <- as.numeric(index\_3)

h\_index <- as.numeric(index\_4)

i\_index <- as.numeric(index\_5)

# Increment the value at the specified position

counts[x\_index, y\_index, z\_index, h\_index, i\_index] <- counts[x\_index, y\_index, z\_index, h\_index, i\_index] + 1

}

library(entropy)

strimmer\_entropy <- entropy(counts)

print(paste("The Shannon entropy calculated via the Strimmer Package is: ",as.character(strimmer\_entropy)))

probs <- counts / sum(counts)

shannon\_entropy <- -sum(probs \* log(probs))

print(paste("The Shannon entropy calculated via the Shannon formula is: ",as.character(shannon\_entropy)))

return (entropy(counts))

}

### Step 3: Use function

q63\_age\_entropy <- Two\_Var\_Entropy(q63\_encoded,age\_encoded)

## [1] "The Shannon entropy calculated via the Strimmer Package is: 3.49053649343863"

## [1] "The Shannon entropy calculated via the Shannon formula is: NaN"

q63\_age\_entropy

## [1] 3.490536

q63\_sex\_age\_entropy <- Three\_Var\_Entropy(q63\_encoded,q146\_encoded,q69\_encoded)

## [1] "The Shannon entropy calculated via the Strimmer Package is: 4.37365254768865"

## [1] "The Shannon entropy calculated via the Shannon formula is: 4.37365254768865"

q63\_sex\_age\_entropy

## [1] 4.373653

q63\_q69\_sex\_age\_entropy <- Four\_Var\_Entropy(q63\_encoded,q69\_encoded,sex\_encoded,age\_encoded)

## [1] "The Shannon entropy calculated via the Strimmer Package is: 5.51544119335772"

## [1] "The Shannon entropy calculated via the Shannon formula is: NaN"

q63\_q69\_sex\_age\_entropy

## [1] 5.515441

q63\_q69\_edu1\_sex\_age\_entropy <- Five\_Var\_Entropy(q63\_encoded,q69\_encoded,edu\_encoded,sex\_encoded,age\_encoded)

## [1] "The Shannon entropy calculated via the Strimmer Package is: 6.7726790476095"

## [1] "The Shannon entropy calculated via the Shannon formula is: NaN"

q63\_q69\_edu1\_sex\_age\_entropy

## [1] 6.772679

q63\_q69\_edu2\_sex\_age\_entropy <- Five\_Var\_Entropy(q63\_encoded,q69\_encoded,edu2\_encoded,sex\_encoded,age\_encoded)

## [1] "The Shannon entropy calculated via the Strimmer Package is: 6.85564778587888"

## [1] "The Shannon entropy calculated via the Shannon formula is: NaN"

q63\_q69\_edu2\_sex\_age\_entropy

## [1] 6.855648

## A summary of all entropies for 5 mixed variables

This function accepts 5 inputs, which are all matrices. Preprocess the variables into matrices before passing variables into the function. Requirements are specified as follows: 1st input - Ordinal, values are in 5 categories. (e.g.Q63) 2st input - Ordinal, values are in 5 categories. (e.g.Q69) 3st input - Ordinal, values are in 5 categories. It can be in distinct or cumulative scheme. (e.g.Education) 4st input - Binary, values are in 0 and 1. (e.g. Sex) 5st input - Continuous, values are in 10 categories. (e.g. Age)

library(entropy)

Entropy\_Summary <- function(r1\_encoded,r2\_encoded,o\_encoded,b\_encoded,c\_encoded){

# Individual Entropies

r1\_counts <- colSums(r1\_encoded)

r2\_counts <- colSums(r2\_encoded)

o\_counts <- colSums(o\_encoded)

b\_counts <- colSums(b\_encoded)

c\_counts <- colSums(c\_encoded)

r1\_entropy <- entropy(r1\_counts)

r2\_entropy <- entropy(r2\_counts)

o\_entropy <- entropy(o\_counts)

b\_entropy <- entropy(b\_counts)

c\_entropy <- entropy(c\_counts)

# Two Joint Entropies

r1\_r2\_entropy <- Two\_Var\_Entropy(r1\_encoded,r2\_encoded)

r1\_o\_entropy <- Two\_Var\_Entropy(r1\_encoded,o\_encoded)

r1\_b\_entropy <- Two\_Var\_Entropy(r1\_encoded,b\_encoded)

r1\_c\_entropy <- Two\_Var\_Entropy(r1\_encoded,c\_encoded)

r2\_o\_entropy <- Two\_Var\_Entropy(r2\_encoded,o\_encoded)

r2\_b\_entropy <- Two\_Var\_Entropy(r2\_encoded,b\_encoded)

r2\_c\_entropy <- Two\_Var\_Entropy(r2\_encoded,c\_encoded)

o\_b\_entropy <- Two\_Var\_Entropy(o\_encoded,b\_encoded)

o\_c\_entropy <- Two\_Var\_Entropy(o\_encoded,c\_encoded)

b\_c\_entropy <- Two\_Var\_Entropy(b\_encoded,c\_encoded)

o\_c\_entropy <- Two\_Var\_Entropy(o\_encoded,c\_encoded)

# Three Joint Entropies

r1\_r2\_o\_entropy <- Three\_Var\_Entropy(r1\_encoded,r2\_encoded,o\_encoded)

r1\_r2\_b\_entropy <- Three\_Var\_Entropy(r1\_encoded,r2\_encoded,b\_encoded)

r1\_r2\_c\_entropy <- Three\_Var\_Entropy(r1\_encoded,r2\_encoded,c\_encoded)

r2\_o\_b\_entropy <- Three\_Var\_Entropy(r2\_encoded,o\_encoded,b\_encoded)

r2\_o\_c\_entropy <- Three\_Var\_Entropy(r2\_encoded,o\_encoded,c\_encoded)

o\_b\_c\_entropy <- Three\_Var\_Entropy(o\_encoded,b\_encoded,c\_encoded)

r1\_b\_c\_entropy <- Three\_Var\_Entropy(r1\_encoded,b\_encoded,c\_encoded)

r2\_b\_c\_entropy <- Three\_Var\_Entropy(r2\_encoded,b\_encoded,c\_encoded)

r1\_o\_b\_entropy <- Three\_Var\_Entropy(r1\_encoded,o\_encoded,b\_encoded)

r1\_o\_c\_entropy <- Three\_Var\_Entropy(r1\_encoded,o\_encoded,c\_encoded)

# Four Joint Entropies

r1\_r2\_o\_b\_entropy <- Four\_Var\_Entropy(r1\_encoded,r2\_encoded,o\_encoded, b\_encoded)

r1\_r2\_o\_c\_entropy <- Four\_Var\_Entropy(r1\_encoded,r2\_encoded,o\_encoded, c\_encoded)

r1\_r2\_b\_c\_entropy <- Four\_Var\_Entropy(r1\_encoded,r2\_encoded,b\_encoded, c\_encoded)

r1\_o\_b\_c\_entropy <- Four\_Var\_Entropy(r1\_encoded,o\_encoded,b\_encoded, c\_encoded)

r2\_o\_b\_c\_entropy <- Four\_Var\_Entropy(r2\_encoded,o\_encoded,b\_encoded, c\_encoded)

# Five Joint Entropies

r1\_r2\_o\_b\_c\_entropy <- Five\_Var\_Entropy(r1\_encoded, r2\_encoded,o\_encoded,b\_encoded, c\_encoded)

# Output in the format of tables

# Individual entropies

one\_output <- c(r1\_entropy, r2\_entropy, o\_entropy, b\_entropy, c\_entropy)

names(one\_output) <- c("r1", "r2", "o", "b", "c")

cat("Individual Entropies:\n")

print(one\_output)

# Two Variables Joint Entropies

two\_output <- matrix('n.a.', nrow = 5, ncol = 5)

rownames(two\_output) <- c("r1", "r2", "o", "b", "c")

colnames(two\_output) <- c("r1", "r2", "o", "b", "c")

two\_output[1, 2] <- r1\_r2\_entropy

two\_output[1, 3] <- r1\_o\_entropy

two\_output[1, 4] <- r1\_b\_entropy

two\_output[1, 5] <- r1\_c\_entropy

two\_output[2, 3] <- r2\_o\_entropy

two\_output[2, 4] <- r2\_b\_entropy

two\_output[2, 5] <- r2\_c\_entropy

two\_output[3, 4] <- o\_b\_entropy

two\_output[3, 5] <- o\_c\_entropy

two\_output[4, 5] <- b\_c\_entropy

cat("\nTwo Variables Joint Entropies:\n")

print(two\_output)

# Three Variables Joint Entropies

# Create a named vector from the matrix

three\_output <- c(

r1\_r2\_o\_entropy,

r1\_r2\_b\_entropy,

r1\_r2\_c\_entropy,

r2\_o\_b\_entropy,

r2\_o\_c\_entropy,

o\_b\_c\_entropy,

r1\_b\_c\_entropy,

r2\_b\_c\_entropy,

r1\_o\_b\_entropy,

r1\_o\_c\_entropy

)

names(three\_output) <- c("r1\_r2\_o", "r1\_r2\_b", "r1\_r2\_c", "r2\_o\_b", "r2\_o\_c", "o\_b\_c", "r1\_b\_c", "r2\_b\_c", "r1\_o\_b", "r1\_o\_c")

cat("\nThree Variables Joint Entropies:\n")

print(three\_output)

# Four Variables Joint Entropies

four\_output <- c(

r1\_r2\_o\_b\_entropy,

r1\_r2\_o\_c\_entropy,

r1\_r2\_b\_c\_entropy,

r1\_o\_b\_c\_entropy,

r2\_o\_b\_c\_entropy

)

names(four\_output) <- c("r1\_r2\_o\_b", "r1\_r2\_o\_c", "r1\_r2\_b\_c", "r1\_o\_b\_c", "r2\_o\_b\_c")

cat("\nFour Variables Joint Entropies:\n")

print(four\_output)

# Five Variables Joint Entropies

cat("\nFive Variables Joint Entropies:\n")

print(r1\_r2\_o\_b\_c\_entropy)

}

Entropy\_Summary(q63\_encoded,q69\_encoded,edu\_encoded,sex\_encoded,age\_encoded)

## [1] "The Shannon entropy calculated via the Strimmer Package is: 2.93951936616594"

## [1] "The Shannon entropy calculated via the Shannon formula is: 2.93951936616594"

## [1] "The Shannon entropy calculated via the Strimmer Package is: 3.06986118709732"

## [1] "The Shannon entropy calculated via the Shannon formula is: 3.06986118709732"

## [1] "The Shannon entropy calculated via the Strimmer Package is: 2.23410188568923"

## [1] "The Shannon entropy calculated via the Shannon formula is: 2.23410188568923"

## [1] "The Shannon entropy calculated via the Strimmer Package is: 3.49053649343863"

## [1] "The Shannon entropy calculated via the Shannon formula is: NaN"

## [1] "The Shannon entropy calculated via the Strimmer Package is: 3.06771322674724"

## [1] "The Shannon entropy calculated via the Shannon formula is: 3.06771322674724"

## [1] "The Shannon entropy calculated via the Strimmer Package is: 2.23231211192909"

## [1] "The Shannon entropy calculated via the Shannon formula is: 2.23231211192909"

## [1] "The Shannon entropy calculated via the Strimmer Package is: 3.49138443117181"

## [1] "The Shannon entropy calculated via the Shannon formula is: NaN"

## [1] "The Shannon entropy calculated via the Strimmer Package is: 2.20359991531507"

## [1] "The Shannon entropy calculated via the Shannon formula is: 2.20359991531507"

## [1] "The Shannon entropy calculated via the Strimmer Package is: 3.38946648728037"

## [1] "The Shannon entropy calculated via the Shannon formula is: NaN"

## [1] "The Shannon entropy calculated via the Strimmer Package is: 2.63269482732853"

## [1] "The Shannon entropy calculated via the Shannon formula is: 2.63269482732853"

## [1] "The Shannon entropy calculated via the Strimmer Package is: 3.38946648728037"

## [1] "The Shannon entropy calculated via the Shannon formula is: NaN"

## [1] "The Shannon entropy calculated via the Strimmer Package is: 4.43920370211378"

## [1] "The Shannon entropy calculated via the Shannon formula is: 4.43920370211378"

## [1] "The Shannon entropy calculated via the Strimmer Package is: 3.61772534119166"

## [1] "The Shannon entropy calculated via the Shannon formula is: 3.61772534119166"

## [1] "The Shannon entropy calculated via the Strimmer Package is: 4.85867982267966"

## [1] "The Shannon entropy calculated via the Shannon formula is: NaN"

## [1] "The Shannon entropy calculated via the Strimmer Package is: 3.73886343667057"

## [1] "The Shannon entropy calculated via the Shannon formula is: 3.73886343667057"

## [1] "The Shannon entropy calculated via the Strimmer Package is: 4.90870499044204"

## [1] "The Shannon entropy calculated via the Shannon formula is: NaN"

## [1] "The Shannon entropy calculated via the Strimmer Package is: 4.05655217026949"

## [1] "The Shannon entropy calculated via the Shannon formula is: NaN"

## [1] "The Shannon entropy calculated via the Strimmer Package is: 4.16720696260158"

## [1] "The Shannon entropy calculated via the Shannon formula is: NaN"

## [1] "The Shannon entropy calculated via the Strimmer Package is: 4.16966708275518"

## [1] "The Shannon entropy calculated via the Shannon formula is: NaN"

## [1] "The Shannon entropy calculated via the Strimmer Package is: 3.73744512063198"

## [1] "The Shannon entropy calculated via the Shannon formula is: 3.73744512063198"

## [1] "The Shannon entropy calculated via the Strimmer Package is: 4.91221446792232"

## [1] "The Shannon entropy calculated via the Shannon formula is: NaN"

## [1] "The Shannon entropy calculated via the Strimmer Package is: 5.09480488155853"

## [1] "The Shannon entropy calculated via the Shannon formula is: NaN"

## [1] "The Shannon entropy calculated via the Strimmer Package is: 6.19967014763764"

## [1] "The Shannon entropy calculated via the Shannon formula is: NaN"

## [1] "The Shannon entropy calculated via the Strimmer Package is: 5.51544119335772"

## [1] "The Shannon entropy calculated via the Shannon formula is: NaN"

## [1] "The Shannon entropy calculated via the Strimmer Package is: 5.55221077271029"

## [1] "The Shannon entropy calculated via the Shannon formula is: NaN"

## [1] "The Shannon entropy calculated via the Strimmer Package is: 5.55460174656555"

## [1] "The Shannon entropy calculated via the Shannon formula is: NaN"

## [1] "The Shannon entropy calculated via the Strimmer Package is: 6.7726790476095"

## [1] "The Shannon entropy calculated via the Shannon formula is: NaN"

## Individual Entropies:

## r1 r2 o b c

## 1.5518190 1.5489958 1.5290336 0.6866834 1.9474124

##

## Two Variables Joint Entropies:

## r1 r2 o b

## r1 "n.a." "2.93951936616594" "3.06986118709732" "2.23410188568923"

## r2 "n.a." "n.a." "3.06771322674724" "2.23231211192909"

## o "n.a." "n.a." "n.a." "2.20359991531507"

## b "n.a." "n.a." "n.a." "n.a."

## c "n.a." "n.a." "n.a." "n.a."

## c

## r1 "3.49053649343863"

## r2 "3.49138443117181"

## o "3.38946648728037"

## b "2.63269482732853"

## c "n.a."

##

## Three Variables Joint Entropies:

## r1\_r2\_o r1\_r2\_b r1\_r2\_c r2\_o\_b r2\_o\_c o\_b\_c r1\_b\_c r2\_b\_c

## 4.439204 3.617725 4.858680 3.738863 4.908705 4.056552 4.167207 4.169667

## r1\_o\_b r1\_o\_c

## 3.737445 4.912214

##

## Four Variables Joint Entropies:

## r1\_r2\_o\_b r1\_r2\_o\_c r1\_r2\_b\_c r1\_o\_b\_c r2\_o\_b\_c

## 5.094805 6.199670 5.515441 5.552211 5.554602

##

## Five Variables Joint Entropies:

## [1] 6.772679

Entropy\_Summary(q63\_encoded,q69\_encoded,edu2\_encoded,sex\_encoded,age\_encoded)

## [1] "The Shannon entropy calculated via the Strimmer Package is: 2.93951936616594"

## [1] "The Shannon entropy calculated via the Shannon formula is: 2.93951936616594"

## [1] "The Shannon entropy calculated via the Strimmer Package is: 3.0511004883198"

## [1] "The Shannon entropy calculated via the Shannon formula is: 3.0511004883198"

## [1] "The Shannon entropy calculated via the Strimmer Package is: 2.23410188568923"

## [1] "The Shannon entropy calculated via the Shannon formula is: 2.23410188568923"

## [1] "The Shannon entropy calculated via the Strimmer Package is: 3.49053649343863"

## [1] "The Shannon entropy calculated via the Shannon formula is: NaN"

## [1] "The Shannon entropy calculated via the Strimmer Package is: 3.0360388143049"

## [1] "The Shannon entropy calculated via the Shannon formula is: 3.0360388143049"

## [1] "The Shannon entropy calculated via the Strimmer Package is: 2.23231211192909"

## [1] "The Shannon entropy calculated via the Shannon formula is: 2.23231211192909"

## [1] "The Shannon entropy calculated via the Strimmer Package is: 3.49138443117181"

## [1] "The Shannon entropy calculated via the Shannon formula is: NaN"

## [1] "The Shannon entropy calculated via the Strimmer Package is: 2.1837424214328"

## [1] "The Shannon entropy calculated via the Shannon formula is: 2.1837424214328"

## [1] "The Shannon entropy calculated via the Strimmer Package is: 3.33986633378162"

## [1] "The Shannon entropy calculated via the Shannon formula is: NaN"

## [1] "The Shannon entropy calculated via the Strimmer Package is: 2.63269482732853"

## [1] "The Shannon entropy calculated via the Shannon formula is: 2.63269482732853"

## [1] "The Shannon entropy calculated via the Strimmer Package is: 3.33986633378162"

## [1] "The Shannon entropy calculated via the Shannon formula is: NaN"

## [1] "The Shannon entropy calculated via the Strimmer Package is: 4.4304025355975"

## [1] "The Shannon entropy calculated via the Shannon formula is: 4.4304025355975"

## [1] "The Shannon entropy calculated via the Strimmer Package is: 3.61772534119166"

## [1] "The Shannon entropy calculated via the Shannon formula is: 3.61772534119166"

## [1] "The Shannon entropy calculated via the Strimmer Package is: 4.85867982267966"

## [1] "The Shannon entropy calculated via the Shannon formula is: NaN"

## [1] "The Shannon entropy calculated via the Strimmer Package is: 3.70652687018504"

## [1] "The Shannon entropy calculated via the Shannon formula is: 3.70652687018504"

## [1] "The Shannon entropy calculated via the Strimmer Package is: 4.85866125608013"

## [1] "The Shannon entropy calculated via the Shannon formula is: NaN"

## [1] "The Shannon entropy calculated via the Strimmer Package is: 4.01005780312285"

## [1] "The Shannon entropy calculated via the Shannon formula is: NaN"

## [1] "The Shannon entropy calculated via the Strimmer Package is: 4.16720696260158"

## [1] "The Shannon entropy calculated via the Shannon formula is: NaN"

## [1] "The Shannon entropy calculated via the Strimmer Package is: 4.16966708275518"

## [1] "The Shannon entropy calculated via the Shannon formula is: NaN"

## [1] "The Shannon entropy calculated via the Strimmer Package is: 3.71858957296879"

## [1] "The Shannon entropy calculated via the Shannon formula is: 3.71858957296879"

## [1] "The Shannon entropy calculated via the Strimmer Package is: 4.87002741552754"

## [1] "The Shannon entropy calculated via the Shannon formula is: NaN"

## [1] "The Shannon entropy calculated via the Strimmer Package is: 5.09329364465249"

## [1] "The Shannon entropy calculated via the Shannon formula is: NaN"

## [1] "The Shannon entropy calculated via the Strimmer Package is: 6.22177209746448"

## [1] "The Shannon entropy calculated via the Shannon formula is: NaN"

## [1] "The Shannon entropy calculated via the Strimmer Package is: 5.51544119335772"

## [1] "The Shannon entropy calculated via the Shannon formula is: NaN"

## [1] "The Shannon entropy calculated via the Strimmer Package is: 5.52864384294184"

## [1] "The Shannon entropy calculated via the Shannon formula is: NaN"

## [1] "The Shannon entropy calculated via the Strimmer Package is: 5.52187218703636"

## [1] "The Shannon entropy calculated via the Shannon formula is: NaN"

## [1] "The Shannon entropy calculated via the Strimmer Package is: 6.85564778587888"

## [1] "The Shannon entropy calculated via the Shannon formula is: NaN"

## Individual Entropies:

## r1 r2 o b c

## 1.5518190 1.5489958 1.5111458 0.6866834 1.9474124

##

## Two Variables Joint Entropies:

## r1 r2 o b

## r1 "n.a." "2.93951936616594" "3.0511004883198" "2.23410188568923"

## r2 "n.a." "n.a." "3.0360388143049" "2.23231211192909"

## o "n.a." "n.a." "n.a." "2.1837424214328"

## b "n.a." "n.a." "n.a." "n.a."

## c "n.a." "n.a." "n.a." "n.a."

## c

## r1 "3.49053649343863"

## r2 "3.49138443117181"

## o "3.33986633378162"

## b "2.63269482732853"

## c "n.a."

##

## Three Variables Joint Entropies:

## r1\_r2\_o r1\_r2\_b r1\_r2\_c r2\_o\_b r2\_o\_c o\_b\_c r1\_b\_c r2\_b\_c

## 4.430403 3.617725 4.858680 3.706527 4.858661 4.010058 4.167207 4.169667

## r1\_o\_b r1\_o\_c

## 3.718590 4.870027

##

## Four Variables Joint Entropies:

## r1\_r2\_o\_b r1\_r2\_o\_c r1\_r2\_b\_c r1\_o\_b\_c r2\_o\_b\_c

## 5.093294 6.221772 5.515441 5.528644 5.521872

##

## Five Variables Joint Entropies:

## [1] 6.855648

# For the Report

# Experiments to validate that the two-variable joint entropies calculated from the Strimmer Entropy package and the shannon formula are the same.

Two\_Var\_Entropy(q63\_encoded,sex\_encoded)

## [1] "The Shannon entropy calculated via the Strimmer Package is: 2.23410188568923"

## [1] "The Shannon entropy calculated via the Shannon formula is: 2.23410188568923"

## [1] 2.234102

Two\_Var\_Entropy(q63\_encoded,q69\_encoded)

## [1] "The Shannon entropy calculated via the Strimmer Package is: 2.93951936616594"

## [1] "The Shannon entropy calculated via the Shannon formula is: 2.93951936616594"

## [1] 2.939519

Two\_Var\_Entropy(q63\_encoded,edu\_encoded)

## [1] "The Shannon entropy calculated via the Strimmer Package is: 3.06986118709732"

## [1] "The Shannon entropy calculated via the Shannon formula is: 3.06986118709732"

## [1] 3.069861

Three\_Var\_Entropy(q63\_encoded,edu\_encoded,sex\_encoded)

## [1] "The Shannon entropy calculated via the Strimmer Package is: 3.73744512063198"

## [1] "The Shannon entropy calculated via the Shannon formula is: 3.73744512063198"

## [1] 3.737445

Four\_Var\_Entropy(q63\_encoded,edu\_encoded,sex\_encoded,age\_encoded)

## [1] "The Shannon entropy calculated via the Strimmer Package is: 5.55221077271029"

## [1] "The Shannon entropy calculated via the Shannon formula is: NaN"

## [1] 5.552211

Five\_Var\_Entropy(q63\_encoded,q69\_encoded,edu\_encoded,sex\_encoded,age\_encoded)

## [1] "The Shannon entropy calculated via the Strimmer Package is: 6.7726790476095"

## [1] "The Shannon entropy calculated via the Shannon formula is: NaN"

## [1] 6.772679

# Experiments to see that if distinctive and cumulative scheme encoding makes a difference to the joint entropy.

# Education-Sex Joint Entropy

Two\_Var\_Entropy(edu\_encoded,sex\_encoded)

## [1] "The Shannon entropy calculated via the Strimmer Package is: 2.20359991531507"

## [1] "The Shannon entropy calculated via the Shannon formula is: 2.20359991531507"

## [1] 2.2036

Two\_Var\_Entropy(edu2\_encoded,sex\_encoded)

## [1] "The Shannon entropy calculated via the Strimmer Package is: 2.1837424214328"

## [1] "The Shannon entropy calculated via the Shannon formula is: 2.1837424214328"

## [1] 2.183742

# Education-Age Joint Entropy

Two\_Var\_Entropy(edu\_encoded,age\_encoded)

## [1] "The Shannon entropy calculated via the Strimmer Package is: 3.38946648728037"

## [1] "The Shannon entropy calculated via the Shannon formula is: NaN"

## [1] 3.389466

Two\_Var\_Entropy(edu2\_encoded,age\_encoded)

## [1] "The Shannon entropy calculated via the Strimmer Package is: 3.33986633378162"

## [1] "The Shannon entropy calculated via the Shannon formula is: NaN"

## [1] 3.339866

# Education-Sex-Age Joint Entropy

Three\_Var\_Entropy(edu\_encoded,sex\_encoded,age\_encoded)

## [1] "The Shannon entropy calculated via the Strimmer Package is: 4.05655217026949"

## [1] "The Shannon entropy calculated via the Shannon formula is: NaN"

## [1] 4.056552

Three\_Var\_Entropy(edu2\_encoded,sex\_encoded,age\_encoded)

## [1] "The Shannon entropy calculated via the Strimmer Package is: 4.01005780312285"

## [1] "The Shannon entropy calculated via the Shannon formula is: NaN"

## [1] 4.010058