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

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_se
x$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:  0.686683
427438679"

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:  0.686683
427438679"
```

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_ed
u$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_e
du$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 Shanno
n formula is: ",as.character(edu_shannon_entropy)))
## [1] "The Shannon entropy of Education in Scheme 1 calculated via the Shannon for
mula is:  1.52903364090424"
print(paste("The Shannon entropy of Education in Scheme 1 calculated via the entrop
y package is: ",as.character(edu_entropy)))
## [1] "The Shannon entropy of Education in Scheme 1 calculated via the entropy pac
kage 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 Shanno
n formula is: ",as.character(edu2_shannon_entropy)))
## [1] "The Shannon entropy of Education in Scheme 2 calculated via the Shannon for
mula is:  1.51114578051434"
print(paste("The Shannon entropy of Education in Scheme 2 calculated via the entrop
y package is: ",as.character(edu2_entropy)))
## [1] "The Shannon entropy of Education in Scheme 2 calculated via the entropy pac
kage 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:  1.947412
38723464"
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:  1.947412
38723464"
```

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: ",a
s.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:  1.551818
99638059"
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:  1.551818
99638059"
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:  1.548995
8219084"
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:  1.548995
8219084"
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:  1.574136
06580525"
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:  1.563843
34764931"
```

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 tab
les
}
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:  2.93
951936616594"

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:  2.93
951936616594"
```

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 th
e 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_i
ndex] + 1
  #print(paste(index_1,index_2, index_3, index_4)) # Check if the indexs are correc
t 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 pack
age 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_ed
u$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_e
du$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_se
x$Sex))))
ordered_unique_values <- sort(unique(sex))
colnames(sex_encoded) <- paste("Sex_", ordered_unique_values, sep = "")
for (i in 1:nrow(df_sex)) {
  value <- as.character(df_sex$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.chara
cter(strimmer_entropy)))

probs <- counts / sum(counts)
shannon_entropy <- -sum(probs * log(probs))
print(paste("The Shannon entropy calculated via the Shannon formula is: ",as.charac
ter(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.chara
cter(strimmer_entropy)))


probs <- counts / sum(counts)
shannon_entropy <- -sum(probs * log(probs))
print(paste("The Shannon entropy calculated via the Shannon formula is: ",as.charac
ter(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.chara
cter(strimmer_entropy)))


probs <- counts / sum(counts)
shannon_entropy <- -sum(probs * log(probs))
print(paste("The Shannon entropy calculated via the Shannon formula is: ",as.charac
ter(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:  3.490536493438
63"
## [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:  4.373652547688
65"
## [1] "The Shannon entropy calculated via the Shannon formula is:  4.3736525476886
5"
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:  5.515441193357
72"
## [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_encode
d,sex_encoded,age_encoded)
## [1] "The Shannon entropy calculated via the Strimmer Package is:  6.772679047609
5"
## [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_encod
ed,sex_encoded,age_encoded)
## [1] "The Shannon entropy calculated via the Strimmer Package is:  6.855647785878
88"
## [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_encode
d, 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:  2.939519366165
94"
## [1] "The Shannon entropy calculated via the Shannon formula is:  2.9395193661659
4"
## [1] "The Shannon entropy calculated via the Strimmer Package is:  3.069861187097
32"
## [1] "The Shannon entropy calculated via the Shannon formula is:  3.0698611870973
2"
## [1] "The Shannon entropy calculated via the Strimmer Package is:  2.234101885689
23"
## [1] "The Shannon entropy calculated via the Shannon formula is:  2.2341018856892
3"
## [1] "The Shannon entropy calculated via the Strimmer Package is:  3.490536493438
63"
## [1] "The Shannon entropy calculated via the Shannon formula is:  NaN"
## [1] "The Shannon entropy calculated via the Strimmer Package is:  3.067713226747
24"
## [1] "The Shannon entropy calculated via the Shannon formula is:  3.0677132267472
4"
## [1] "The Shannon entropy calculated via the Strimmer Package is:  2.232312111929
09"
## [1] "The Shannon entropy calculated via the Shannon formula is:  2.2323121119290
9"
## [1] "The Shannon entropy calculated via the Strimmer Package is:  3.491384431171
81"
## [1] "The Shannon entropy calculated via the Shannon formula is:  NaN"
## [1] "The Shannon entropy calculated via the Strimmer Package is:  2.203599915315
07"
## [1] "The Shannon entropy calculated via the Shannon formula is:  2.2035999153150
7"
## [1] "The Shannon entropy calculated via the Strimmer Package is:  3.389466487280
37"
## [1] "The Shannon entropy calculated via the Shannon formula is:  NaN"
## [1] "The Shannon entropy calculated via the Strimmer Package is:  2.632694827328
53"
```

```
## [1] "The Shannon entropy calculated via the Shannon formula is:  2.6326948273285
3"

## [1] "The Shannon entropy calculated via the Strimmer Package is:  3.389466487280
37"

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

## [1] "The Shannon entropy calculated via the Strimmer Package is:  4.439203702113
78"

## [1] "The Shannon entropy calculated via the Shannon formula is:  4.4392037021137
8"

## [1] "The Shannon entropy calculated via the Strimmer Package is:  3.617725341191
66"

## [1] "The Shannon entropy calculated via the Shannon formula is:  3.6177253411916
6"

## [1] "The Shannon entropy calculated via the Strimmer Package is:  4.858679822679
66"

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

## [1] "The Shannon entropy calculated via the Strimmer Package is:  3.738863436670
57"

## [1] "The Shannon entropy calculated via the Shannon formula is:  3.7388634366705
7"

## [1] "The Shannon entropy calculated via the Strimmer Package is:  4.908704990442
04"

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

## [1] "The Shannon entropy calculated via the Strimmer Package is:  4.056552170269
49"

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

## [1] "The Shannon entropy calculated via the Strimmer Package is:  4.167206962601
58"

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

## [1] "The Shannon entropy calculated via the Strimmer Package is:  4.169667082755
18"

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

## [1] "The Shannon entropy calculated via the Strimmer Package is:  3.737445120631
98"

## [1] "The Shannon entropy calculated via the Shannon formula is:  3.7374451206319
8"

## [1] "The Shannon entropy calculated via the Strimmer Package is:  4.912214467922
32"

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

## [1] "The Shannon entropy calculated via the Strimmer Package is:  5.094804881558
53"

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

## [1] "The Shannon entropy calculated via the Strimmer Package is:  6.199670147637
64"

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

## [1] "The Shannon entropy calculated via the Strimmer Package is:  5.515441193357
72"
```

```
## [1] "The Shannon entropy calculated via the Shannon formula is:  NaN"
## [1] "The Shannon entropy calculated via the Strimmer Package is:  5.552210772710
29"
## [1] "The Shannon entropy calculated via the Shannon formula is:  NaN"
## [1] "The Shannon entropy calculated via the Strimmer Package is:  5.554601746565
55"
## [1] "The Shannon entropy calculated via the Shannon formula is:  NaN"
## [1] "The Shannon entropy calculated via the Strimmer Package is:  6.772679047609
5"
## [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:  2.939519366165
94"

## [1] "The Shannon entropy calculated via the Shannon formula is:  2.9395193661659
4"

## [1] "The Shannon entropy calculated via the Strimmer Package is:  3.051100488319
8"

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

## [1] "The Shannon entropy calculated via the Strimmer Package is:  2.234101885689
23"

## [1] "The Shannon entropy calculated via the Shannon formula is:  2.2341018856892
3"

## [1] "The Shannon entropy calculated via the Strimmer Package is:  3.490536493438
63"

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

## [1] "The Shannon entropy calculated via the Strimmer Package is:  3.036038814304
9"

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

## [1] "The Shannon entropy calculated via the Strimmer Package is:  2.232312111929
09"

## [1] "The Shannon entropy calculated via the Shannon formula is:  2.2323121119290
9"

## [1] "The Shannon entropy calculated via the Strimmer Package is:  3.491384431171
81"

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

## [1] "The Shannon entropy calculated via the Strimmer Package is:  2.183742421432
8"

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

## [1] "The Shannon entropy calculated via the Strimmer Package is:  3.339866333781
62"

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

## [1] "The Shannon entropy calculated via the Strimmer Package is:  2.632694827328
53"

## [1] "The Shannon entropy calculated via the Shannon formula is:  2.6326948273285
3"

## [1] "The Shannon entropy calculated via the Strimmer Package is:  3.339866333781
62"

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

## [1] "The Shannon entropy calculated via the Strimmer Package is:  4.430402535597
5"

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

## [1] "The Shannon entropy calculated via the Strimmer Package is:  3.617725341191
66"

## [1] "The Shannon entropy calculated via the Shannon formula is:  3.6177253411916
6"
```

```
## [1] "The Shannon entropy calculated via the Strimmer Package is:  4.858679822679
66"

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

## [1] "The Shannon entropy calculated via the Strimmer Package is:  3.706526870185
04"

## [1] "The Shannon entropy calculated via the Shannon formula is:  3.7065268701850
4"

## [1] "The Shannon entropy calculated via the Strimmer Package is:  4.858661256080
13"

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

## [1] "The Shannon entropy calculated via the Strimmer Package is:  4.010057803122
85"

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

## [1] "The Shannon entropy calculated via the Strimmer Package is:  4.167206962601
58"

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

## [1] "The Shannon entropy calculated via the Strimmer Package is:  4.169667082755
18"

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

## [1] "The Shannon entropy calculated via the Strimmer Package is:  3.718589572968
79"

## [1] "The Shannon entropy calculated via the Shannon formula is:  3.7185895729687
9"

## [1] "The Shannon entropy calculated via the Strimmer Package is:  4.870027415527
54"

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

## [1] "The Shannon entropy calculated via the Strimmer Package is:  5.093293644652
49"

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

## [1] "The Shannon entropy calculated via the Strimmer Package is:  6.221772097464
48"

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

## [1] "The Shannon entropy calculated via the Strimmer Package is:  5.515441193357
72"

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

## [1] "The Shannon entropy calculated via the Strimmer Package is:  5.528643842941
84"

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

## [1] "The Shannon entropy calculated via the Strimmer Package is:  5.521872187036
36"

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

## [1] "The Shannon entropy calculated via the Strimmer Package is:  6.855647785878
88"

## [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:  2.234101885689
23"

## [1] "The Shannon entropy calculated via the Shannon formula is:  2.2341018856892
3"

## [1] 2.234102

Two_Var_Entropy(q63_encoded,q69_encoded)
```

```
## [1] "The Shannon entropy calculated via the Strimmer Package is:  2.939519366165
94"

## [1] "The Shannon entropy calculated via the Shannon formula is:  2.9395193661659
4"

## [1] 2.939519

Two_Var_Entropy(q63_encoded,edu_encoded)

## [1] "The Shannon entropy calculated via the Strimmer Package is:  3.069861187097
32"

## [1] "The Shannon entropy calculated via the Shannon formula is:  3.0698611870973
2"

## [1] 3.069861

Three_Var_Entropy(q63_encoded,edu_encoded,sex_encoded)

## [1] "The Shannon entropy calculated via the Strimmer Package is:  3.737445120631
98"

## [1] "The Shannon entropy calculated via the Shannon formula is:  3.7374451206319
8"

## [1] 3.737445

Four_Var_Entropy(q63_encoded,edu_encoded,sex_encoded,age_encoded)

## [1] "The Shannon entropy calculated via the Strimmer Package is:  5.552210772710
29"

## [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:  6.772679047609
5"

## [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 dif
ference 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:  2.203599915315
07"

## [1] "The Shannon entropy calculated via the Shannon formula is:  2.2035999153150
7"

## [1] 2.2036

Two_Var_Entropy(edu2_encoded,sex_encoded)

## [1] "The Shannon entropy calculated via the Strimmer Package is:  2.183742421432
8"

## [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
```