Automation of Inter-Rater Reliability (IRR)

Ailbhe N. Finnerty

04 November2019

Markdown for "IRR_Automation.R"

This is an R Markdown document to provide instructions on how to run the R script **https://github.com/HumanBehaviourChangeProject/Automation-InterRater-Reliability/blob/master/Automation_IRR_Krippendorf.R** to automate the calculation of Inter-rater reliability (IRR) using R package **'irr'**. This work is part of the Human Behaviour Change Project https://www.humanbehaviourchange.org/. More information about the methods used to development this script is currently being written up and will be available online by January 2020.

The input for this script is one or more csv files with at least three columns, if more than one file they must match in both name and number of columns. The basic necessary data structure is **1)** a column for the **entity/attribute** that you want to calculate IRR for, **2)** a column of binary data from Coder 1 and **3)** a column of binary data for Coder 2.

The data used in this example are two csv files ("Behaviour1.csv" and "Behaviour2.csv") from a parsed JSON file of annotations created by EPPI reviewer **https://eppi.ioe.ac.uk/cms/**. The data is parsed from the JSON files using a python script which can be found at **https://github.com/HumanBehaviourChangeProject/Automation-InterRater-Reliability/blob/master/IrrKrippendorf.py**.

Step 1 - Getting setup

The first step is to save the "IRR_Automation.R" script and your xslx or csv files in the same folder on your computer and set this folder as your working directory.

```
# dir<-"C:/Users/Desktop/Automation IRR"
# setwd(dir)
```

You then need to install the required packages to run the script

```
# install.packages('irr')
# install.packages('data.table')
# install.packages('plyr')
# install.packages('dplyr')
# install.packages('reshape2')
# install.packages('splitstackshape')
# install.packages('tidyverse')
#install.packages('tinytex')
```

and load the installed packages.

```
library(irr)
```

```
## Warning: package 'irr' was built under R version 3.5.3
```

```
## Loading required package: lpSolve
```

```
## Warning: package 'lpSolve' was built under R version 3.5.2
```

```
# library(data.table)
 library(plyr)
```

```
## Warning: package 'plyr' was built under R version 3.5.3
```

```
# library(dplyr)
# library(reshape2)
# library(splitstackshape)
# library(tidyverse)
#library(tinytex)
```

Step 2 - Loading the data

Once the working directory has been established and you have loaded the required packages to run the script, the next step is to load the data. We are using two csv files that contain binary data from two coders which has been parsed from JSON files by the python script which can be found here **https://github.com/HumanBehaviourChangeProject/Automation-InterRater-Reliability**. In this step we load the two separate files and as they had the same format and headings we could combine them as a 'data.frame' using 'rbind'. We used 'na.omit' to ensure that there is no missing data with listwise deletion of missing values.

```
File1<-read.csv("Behaviour1.csv", header = TRUE,sep = ",")
File2<-read.csv("Behaviour2.csv", header = TRUE,sep = ",")
DataFile<-rbind.data.frame(File1, File2)
DataFile<-na.omit(DataFile)
head(DataFile, n=5)
```

```
##    paperID     shortTitle AttributeId                    AttributeTitle
## 1 34829909 Aveyard (2001)     4788957              Geographical location
## 2 34829909 Aveyard (2001)     4788958           Country of intervention
## 3 34829909 Aveyard (2001)     4788959    Lower-level geographical region
## 4 34829909 Aveyard (2001)     5295657              Attribute of location
## 5 34829909 Aveyard (2001)     4788960 Area social and economic condition
##      ArmTitle Coder1Text Coder2Text
## 1 Whole Study          0          0
## 2 Whole Study          1          1
## 3 Whole Study          1          1
## 4 Whole Study          0          0
## 5 Whole Study          0          0
```

```
names(DataFile)
```

```
## [1] "paperID"        "shortTitle"     "AttributeId"    "AttributeTitle"
## [5] "ArmTitle"       "Coder1Text"     "Coder2Text"
```

Step 3 - Restructuring the data

The DataFile data.frame has more columns of data than we need to calculate *krippendorf's alpha*. We want to calculate the agreement between Coder1 (Coder1Text) and Coder2 (Coder2Text) for each attribute (AttributeID/AttributeTitle) in all papers (shortTitle) that were annotated.

```
DataFile_Attributes<-DataFile[,c((4),(2:3),(6:7))]
head(DataFile_Attributes, n=5)
```

```
##                       AttributeTitle     shortTitle AttributeId Coder1Text
## 1              Geographical location Aveyard (2001)     4788957          0
## 2            Country of intervention Aveyard (2001)     4788958          1
## 3    Lower-level geographical region Aveyard (2001)     4788959          1
## 4              Attribute of location Aveyard (2001)     5295657          0
## 5 Area social and economic condition Aveyard (2001)     4788960          0
##   Coder2Text
## 1          0
## 2          1
## 3          1
## 4          0
## 5          0
```

We want to check how many papers were annotated and how many attributes we want to calculate the statistic for.

```
NumPapers<-unique(DataFile_Attributes$shortTitle)
NumEntities<-unique(DataFile_Attributes$AttributeTitle)
length(NumPapers)
```

```
## [1] 50
```

```
length(NumEntities)
```

```
## [1] 70
```

This tells us that we have annotated data for 50 papers and for 70 unique attributes.

Step 4 - Finding attributes with data

We want to calcualate IRR for the entire dataset and for each attribute but as there are many entities we want to check if any of them have not been annotated at all for the 50 papers to exclude them from the final analysis. To check this we simply count the annotations for each attribute.

```
AttributeCount<-ddply(DataFile_Attributes, "AttributeTitle", numcolwise(sum))
tail(AttributeCount)
```

```
##         AttributeTitle AttributeId Coder1Text Coder2Text
## 65 Temporary residence   241468730          0          0
## 66      Transportation   241471940          0          0
## 67 University facility   241471100          0          2
## 68          Urban area   241468250          4          4
## 69 Vocational facility   246361044          2          2
## 70               Water   247280720          0          0
```

This allows us to find the attributes that have data for at least one coder and those that have no data at all for either coder 1 or coder 2.

```
HasData<-as.data.frame(AttributeCount[ which(AttributeCount$Coder1Text > 0 | AttributeCount$Coder2Text >
HasNoData<-as.data.frame(AttributeCount[ which(AttributeCount$Coder1Text == 0 & AttributeCount$Coder2Te
length(unique(HasData$AttributeTitle))
```

```
## [1] 24
```

```
print(HasData)
```

```
##                          AttributeTitle AttributeId Coder1Text Coder2Text
## 8         Community healthcare facility   264782910          1          3
## 10              Country of intervention   241468000         20         15
## 13    Doctor-led primary care facility   241469270          9          7
## 17       Emergency department facility   241470750          1          1
## 18                            Facility   241468350          3          1
## 23                 Healthcare facility   241468780          3          4
## 27                   Hospital facility   241468830          1          1
## 28 Hospital outpatient clinic facility   241470620          4          4
## 30                 Household residence   241468480          2          1
## 33     Lower-level geographical region   241468050         33         26
## 38                     Office facility   241471630          0          1
## 39                 Outdoor environment   246362154          1          1
## 46                      Primary school   241470950          1          1
## 48                Psychiatric facility   241610380          1          1
## 52                   Research facility   246260630          3          2
## 54                Residential facility   241468430          1          1
## 57                           Rural area   241468200          2          1
## 58                      School facility   241470900          2          1
## 59                    Secondary school   241471000          5          5
## 62         Sport and exercise facility   246361248          1          1
## 64                       Suburban area   264782840          3          3
## 67                 University facility   241471100          0          2
## 68                           Urban area   241468250          4          4
## 69                 Vocational facility   246361044          2          2
```

Now that we know there are only 24 attributes with data we can subset the original data.frame so that it includes attributes with data only and order the data.frame by the headers of AttributeTitle and shortTitle.

```
NewData<-match_df(DataFile_Attributes, HasData, on = "AttributeTitle")
NewData<-NewData[order(NewData$AttributeTitle, NewData$shortTitle),]
```

Step 5 - Calculating IRR using kripp.alpha

We can now calculate *krippendorf's alpha* for the entire data.frame with and without data. The 'kripp.alpha' function requires the data to be in a table format and calculates IRR for all columns. We further subset the data to just include the columns with data for Coder1 and Coder2.

```
All_Attributes<-t(DataFile_Attributes[,4:5])
Result<-kripp.alpha(All_Attributes, method = c("ordinal"))
Statistic<-as.character("All Entities")
Value<-as.numeric(Result$value)
All_Data_Result<-data.frame(Statistic,Value)
print(All_Data_Result)
```

```
##      Statistic     Value
## 1 All Entities 0.7322859
```

We are more interested in the alpha value for the attributes with data only as we are assuming that the value for the entire dataset will be inflated by the attributes with no data for either coder as they would technically have perfect agreeement. For this reason we do a second calcuation on the subset of data with attributes.

```
Attributes_with_Data<-t(NewData[,4:5])
Attributes_with_Data<-na.omit(Attributes_with_Data)
Result<-kripp.alpha(Attributes_with_Data, method = c("ordinal"))
Statistic<-as.character("Entities with data")
Value<-as.numeric(Result$value)
Attribute_Result<-data.frame(Statistic,Value)
print(Attribute_Result)
```

```
##            Statistic     Value
## 1 Entities with data 0.7171377
```

Our final calculation is for each of the attributes indiviually, with or without data, which requires us to loop over each of the attributes in turn. we want to print all the results in a new data.frame 'all_results'.

Our for loop creates a data.frame with the values for Coder1 and Coder2 only for each attribute in turn and calculates kripp.alpha for each attribute. As we loop over each attribute we need to save the alpha values storing the values in a temporary 'new_data_frame' before moving onto the next attribute.

```
all_results<-data.frame()

for (attributetitle in unique(NewData$AttributeTitle)) {

  temp_data = NewData[NewData$AttributeTitle==attributetitle,]
  temp_data$Coder1Text<-as.numeric(temp_data$Coder1Text)
  temp_data$Coder2Text<-as.numeric(temp_data$Coder2Text)

  temp_data2<-t(temp_data[4:5])

  result<-kripp.alpha(temp_data2, method = c("ordinal"))
  value<-as.numeric(result$value)

  new_data_frame = data.frame(AttributeTitle=attributetitle,Result=value)

  all_results <- rbind(all_results,new_data_frame)
}
```

We then join all the values together in our 'all_results' data.frame.

```
all_results <- rbind(all_results,new_data_frame)
```

We want to create an output which will have all the alpha scores which we calculated, for the whole dataset, the attributes with data subset and the individual entities and so we first join the two alphas score and rename the column headers to match with the 'all_results' data.frame.

```
WholeSample<-rbind(All_Data_Result,Attribute_Result)
names(WholeSample) <- c("AttributeTitle", "Result")
```

We then add both data.frames together.

```
AllResults <-rbind(all_results, WholeSample)
head(AllResults, n=10)
```

```
##                          AttributeTitle      Result
## 1         Community healthcare facility  0.48437500
## 2               Country of intervention  0.78241758
## 3      Doctor-led primary care facility  0.70535714
## 4         Emergency department facility  1.00000000
## 5                               Facility  0.48437500
## 6                   Healthcare facility  0.23963134
## 7                     Hospital facility  1.00000000
## 8  Hospital outpatient clinic facility  0.73097826
## 9                   Household residence -0.02061856
## 10     Lower-level geographical region  0.71351798
```

Finally we write the output to a csv.file.

```
write.csv(AllResults, "IRR_Results_AlphaValues.csv")
```