Kathryn Phillips

Macquarie University 43686826

**FOAR705 Proof of Concept code repository**

Program used: R-3.3.3 (64-bit)

**Code instructions**

1. Download the .csv file “names\_removed.csv” and place it in a directory.
2. Make sure R is installed and set up. Ensure that the data.table package is installed as well- often it is not included in initial R setup.
   * If data.table is not installed in the same location as the rest of the R, create a new directory and install the package from there when you first run R.

**Eg:**

> install.packages("data.table",lib="/R/win-library/3.3")

Warning in install.packages("data.table", lib = "/R/win-library/3.3") :

'lib = "/R/win-library/3.3"' is not writable

--- Please select a CRAN mirror for use in this session ---

trying URL 'https://cran.csiro.au/bin/windows/contrib/3.3/data.table\_1.10.4.zip'

Content type 'application/zip' length 1502201 bytes (1.4 MB)

downloaded 1.4 MB

package ‘data.table’ successfully unpacked and MD5 sums checked

1. Run R
2. Check .csv file is in the right working directory:

> getwd()

1. If need be, set the working directory to new location to access the .csv file:

> setwd("directory address")

**Note:** in this case “directory address” should be the location where you have placed the .csv file. Eg: “C:/Users/Katy/Documents/”.

1. Load .csv and name it for easy access within R.

Name<-read.csv(“Name.csv”,header=TRUE/FALSE,stringsasfactors=TRUE/FALSE)

**Note:** header=TRUE/FALSE is whether there is a header or not TRUE = yes, FALSE = no

stringsAsFactors is linked to the csv file. Usually needs to be set to FALSE (stringsAsFactors relates to whether the strings in the data should be treated as factor variables for modelling functions)

1. Print the full csv out in R by using the shortcut you just established (aka ***Name***<-read.csv())
2. Once loaded, separate all the data within each question column. Since a data table cannot work with more than one piece of data in each row, it is best to have a cleaned version of the columns where more than one option was available (in names\_removed.csv, these columns would be **Genres Listened To**, and **Pop Sub-genres)**

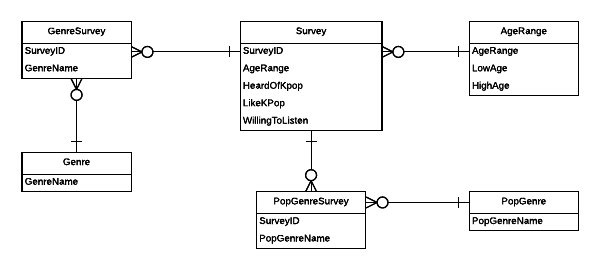
**Note:** For ease of reference, I wrote out my lists of information within a notebook. This way I could refer to the data while constructing my tables without having to constantly switch windows of work on my computer. This window-switching can also be countered by using a split-screen or more than one monitor with your computer.

1. Construct a data table plan for storing the information. In order to ask multiple questions of your data and comply with 3NF (3rd Normal Form), you will need to normalise this table.

**There are numerous “Normal Forms”. Here are the three definitions you need to understand for this task:**

* 1. **First Normal Form (1NF)**
     + Data is stored in a data table
     + Information stored in rows, where at least one of the columns known as the “primary key” identifies each of the rows
     + There are no sub-columns
  2. **Second Normal Form (2NF)**
     + The table is in first normal form
     + All of the non-key components are defined by the primary key
  3. **Third Normal Form (3NF)**
     + The table is in second normal form
     + It contains only columns that are non-transitively dependent on the primary key (the columns cannot rely on any intermediary column as well as the primary key to give it value)

Since normalisation was outside of my proof of concept requirements, I received assistance structuring my data into the form below:



Using more than one box for all the data satisfied the requirements of **3NF**. When I needed to analyse data on Genre, Pop Genre or Age, I could perform an “inner join” of the tables to create a new table. This is explained in later steps.

**ATTENTION:** The order of construction for this database is **VERY IMPORTANT**. Please construct the tables in the following order.

1. Genre
2. PopGenre
3. AgeRange
4. Survey
5. GenreSurvey
6. PopGenreSurvey

**Code (and results from code):**

> Genre<-data.table(GenreName=c("Rock","Classical","Heavy Metal","Videogame Music","Hip Hop","Folk Music","Jazz","Pop","Musicals","Punk","World Music","Country","Choral"))

> Genre

GenreName

1: Choral

2: Classical

3: Country

4: Folk Music

5: Heavy Metal

6: Hip Hop

7: Jazz

8: Musicals

9: Pop

10: Punk

11: Rock

12: Videogame Music

13: World Music

> PopGenre<-data.table(PopGenreName=c("N/A","Art Pop","Experimental Pop","Indie Pop","Pop Punk","Pop Rock","Asian Pop","Bubblegum Pop"))

> PopGenre

PopGenreName

1: N/A

2: Art Pop

3: Experimental Pop

4: Indie Pop

5: Pop Punk

6: Pop Rock

7: Asian Pop

8: Bubblegum Pop

> AgeRange<-data.table(AgeRange=c("18-29","30-39","40-49","50-59","60+"),LowAge=c("18","30","40","50","60"),HighAge=c("29","39","49","59","."))

> AgeRange

AgeRange LowAge HighAge

1: 18-29 18 29

2: 30-39 30 39

3: 40-49 40 49

4: 50-59 50 59

5: 60+ 60 .

NB: The . in the High Age means that there is no upper limit to the age group “60+”

> Survey<-data.table(ID=c("1","2","3","4","5","6","7"),AgeRange=c("18-29","18-29","40-49","30-39","30-39","18-29","50-59"),HaveYouHeardOfKPop=c("No","Yes","No","Yes","Yes","Yes","Yes"),DoYouLikeKPop=c("N/A","No","N/A","Yes","No","Yes","Yes"),WouldYouListenToKpop=c("Yes","No","Maybe","Yes","Maybe","N/A","Yes"))

>Survey

ID AgeRange HaveYouHeardOfKPop DoYouLikeKPop WouldYouListenToKpop

1: 1 18-29 No N/A Yes

2: 2 18-29 Yes No No

3: 3 40-49 No N/A Maybe

4: 4 30-39 Yes Yes Yes

5: 5 30-39 Yes No Maybe

6: 6 18-29 Yes Yes N/A

7: 7 50-59 Yes Yes Yes

> GenreSurvey<-data.table(ID=c("1","1","1","1","2","2","3","3","3","3","4","4","4","5","5","5","6","6","6","7","7","7","7","7","7","7","7"),GenreName=c("Rock","Classical","Heavy Metal","Videogame Music","Rock","Hip Hop","Folk Music","Hip Hop","Jazz","Pop","Classical","Musicals","Punk","Rock","Pop","Heavy Metal","Rock","Hip Hop","World Music","Rock","Classical","Folk Music","Blues","Jazz","World Music","Country","Choral"))

> GenreSurvey

ID GenreName

1: 1 Rock

2: 1 Classical

3: 1 Heavy Metal

4: 1 Videogame Music

5: 2 Rock

6: 2 Hip Hop

7: 3 Folk Music

8: 3 Hip Hop

9: 3 Jazz

10: 3 Pop

11: 4 Classical

12: 4 Musicals

13: 4 Punk

14: 5 Rock

15: 5 Pop

16: 5 Heavy Metal

17: 6 Rock

18: 6 Hip Hop

19: 6 World Music

20: 7 Rock

21: 7 Classical

22: 7 Folk Music

23: 7 Blues

24: 7 Jazz

25: 7 World Music

26: 7 Country

27: 7 Choral

ID GenreName

> PopGenreSurvey<-data.table(ID=c(1,2,3,3,3,4,5,5,5,6,6,6,7),PopGenreName=c("N/A","N/A","Art Pop","Experimental Pop","Indie Pop","N/A","Indie Pop","Pop Punk","Pop Rock","Asian Pop","Bubblegum Pop","Indie Pop","N/A"))

> PopGenreSurvey

ID PopGenreName

1: 1 N/A

2: 2 N/A

3: 3 Art Pop

4: 3 Experimental Pop

5: 3 Indie Pop

6: 4 N/A

7: 5 Indie Pop

8: 5 Pop Punk

9: 5 Pop Rock

10: 6 Asian Pop

11: 6 Bubblegum Pop

12: 6 Indie Pop

13: 7 N/A

1. Save all of these tables as .csv files for future use if necessary:

> write.csv(Genre,file="Genre.csv")

> write.csv(PopGenre,file="PopGenre.csv")

> write.csv(AgeRange,file="AgeRange.csv")

> write.csv(Survey,file="Survey.csv")

> write.csv(GenreSurvey,file="GenreSurvey.csv")

> write.csv(PopGenreSurvey,file="PopGenreSurvey.csv")

1. Decide what your question is. If you need to use multiple tables, perform an inner join to connect the necessary ones:

**Note:** an inner join means that two tables can be joined together to make a new one. Using a key (usually a chosen column title that is found in both tables) you can create a table where each matched pair in the two table’s rows are combined.

E.g: In my proof of concept, I chose to answer the question: What Genres are the most popular among my 18-29 year old respondents?

Therefore, I join together Genre, GenreSurvey and Survey to produce the table I need.

**Code for the inner joins:**

> setkey(Genre,GenreName)

> setkey(GenreSurvey,GenreName)

> GenreJoin<-GenreSurvey[Genre,nomatch=0]

> GenreJoin

ID GenreName

1: 7 Choral

2: 1 Classical

3: 4 Classical

4: 7 Classical

5: 7 Country

6: 3 Folk Music

7: 7 Folk Music

8: 1 Heavy Metal

9: 5 Heavy Metal

10: 2 Hip Hop

11: 3 Hip Hop

12: 6 Hip Hop

13: 3 Jazz

14: 7 Jazz

15: 4 Musicals

16: 3 Pop

17: 5 Pop

18: 4 Punk

19: 1 Rock

20: 2 Rock

21: 5 Rock

22: 6 Rock

23: 7 Rock

24: 1 Videogame Music

25: 6 World Music

26: 7 World Music

ID GenreName

> setkey(GenreJoin,ID)

> setkey(Survey,ID)

> SurveyGenreJoin<-Survey[GenreJoin,nomatch=0]

> SurveyGenreJoin

ID AgeRange HaveYouHeardOfKPop|DoYouLikeKPop WouldYouListenToKpop |GenreName

1: 1 18-29 No N/A Yes Classical

2: 1 18-29 No N/A Yes Heavy Metal

3: 1 18-29 No N/A Yes Rock

4: 1 18-29 No N/A Yes Videogame Music

5: 2 18-29 Yes No No Hip Hop

6: 2 18-29 Yes No No Rock

7: 3 40-49 No N/A Maybe Folk Music

8: 3 40-49 No N/A Maybe Hip Hop

9: 3 40-49 No N/A Maybe Jazz

10: 3 40-49 No N/A Maybe Pop

11: 4 30-39 Yes Yes Yes Classical

12: 4 30-39 Yes Yes Yes Musicals

13: 4 30-39 Yes Yes Yes Punk

14: 5 30-39 Yes No Maybe Heavy Metal

15: 5 30-39 Yes No Maybe Pop

16: 5 30-39 Yes No Maybe Rock

17: 6 18-29 Yes Yes N/A Hip Hop

18: 6 18-29 Yes Yes N/A Rock

19: 6 18-29 Yes Yes N/A World Music

20: 7 50-59 Yes Yes Yes Choral

21: 7 50-59 Yes Yes Yes Classical

22: 7 50-59 Yes Yes Yes Country

23: 7 50-59 Yes Yes Yes Folk Music

24: 7 50-59 Yes Yes Yes Jazz

25: 7 50-59 Yes Yes Yes Rock

26: 7 50-59 Yes Yes Yes World Music

ID AgeRange HaveYouHeardOfKPop|DoYouLikeKPop| WouldYouListenToKpop|GenreName

1. Once all the inner joins have been completed, save the new survey as a .csv – you can always come back to it later and will cut out the hassle of re-joining the code together.

> write.csv(SurveyGenreJoin,"SurveyGenreJoin.csv")

1. Now, select the subset you wish to examine. In this case my subset was the 18-29 age group.

> DemographicTest<-SurveyGenreJoin[AgeRange=="18-29"]

> DemographicTest

ID AgeRange HaveYouHeardOfKPop DoYouLikeKPop WouldYouListenToKpop GenreName

1: 1 18-29 No N/A Yes Classical

2: 1 18-29 No N/A Yes Heavy Metal

3: 1 18-29 No N/A Yes Rock

4: 1 18-29 No N/A Yes Videogame Music

5: 2 18-29 Yes No No Hip Hop

6: 2 18-29 Yes No No Rock

7: 6 18-29 Yes Yes N/A Hip Hop

8: 6 18-29 Yes Yes N/A Rock

9: 6 18-29 Yes Yes N/A World Music

1. Once I had my demographic, I wanted to only look at what genres they listened to. I narrowed down the table yet again.

> DemographicTest[,.(GenreName)]

GenreName

1: Classical

2: Heavy Metal

3: Rock

4: Videogame Music

5: Hip Hop

6: Rock

7: Hip Hop

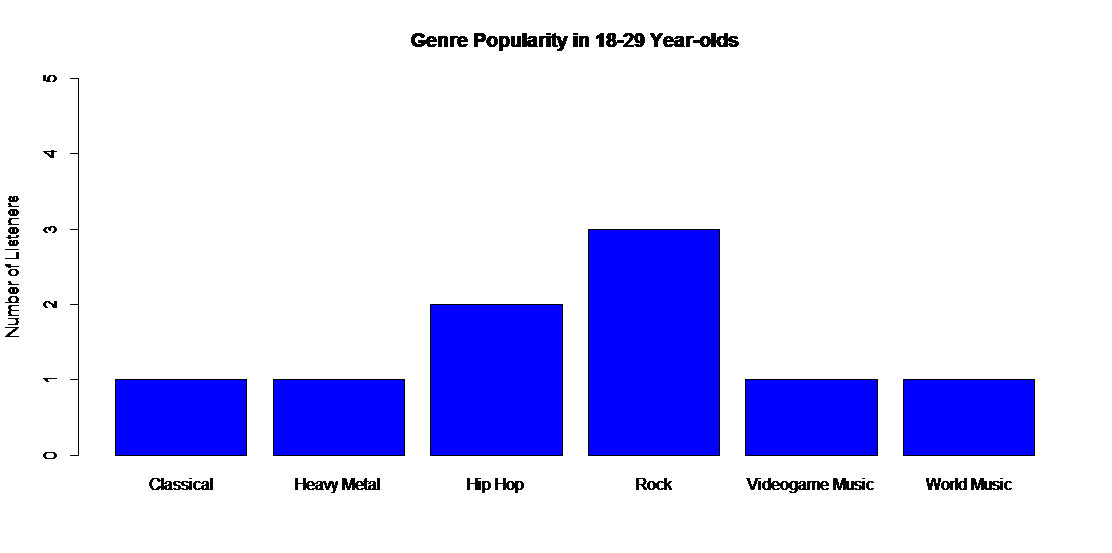
8: Rock

9: World Music

1. From here, I wanted to make a bar chart.

> barplot(counts,main="Genre Popularity in 18-29 Year-olds",ylab="Number of Listeners",ylim=c(0,5),col=c("blue"))

1. This came up with the following successful chart:



1. To import it into word, I saved the image using the following steps:

Right click → Save as metafile→ Choose working directory and name the file using a .png/.jpeg ID→ toggle the file type to all files → Click ‘Save’

1. The code up to this point has proven my Proof of concept (taking raw .csv data from a survey, being able to analyse a specific section of it, then create a graphical image that can be imported into a word processing document/other document. However, if you wish to try out other codes for this set of data, please follow these instructions and make use of the PDF links available, which provide clear examples of simple codes to use and their intended effects.

**Additional code:**

>plot(x) – creates a simple histogram chart of data from graph “x”

>X<–c(a,b,c) –creates a vector of a,b,c , and labels it as X ready for quick retrieval and use in analysis.

> table((DemographicTest[,.(GenreName)])) – finds the frequency of answers within the table “DemographicTest”. The [.,(GenreName)] means that within the table only the answers for the column “GenreName” are examined. “.,” means that there is no row value to search for when choosing the data to count.

>pie(Slices,labels=Labels,main=”Title”) – creates a pie-chart from the data you choose.

**NOTE:** In order to correctly produce a pie-chart, you must establish a vector of the frequencies of responses for each option and label it as “Slices”, and create a vector for the lables as well, naming it “Labels” or something similar. The main=”Title” is the main title of the graph, where “Title” can be changed to a suitable heading.

**Chart/graph additions:**

col=rainbow/red/blue etc. – changes the colour of the chart being created. Place inside the brackets containing the data. Eg: >pie(Slices,labels=Labels,main=”Title”,col=rainbow)

pie3D() – changes a normal pie chart to a 3D pie chart.

Explode=0.1 – placed within brackets in a pie chart. It separates the segments of the chart by the scale written (the =0.1 in the example above

**Useful websites and sources I used for my code and skill acquisition:**

<http://www.statmethods.net/index.html> - Quick-R: a well-rounded site with instructions on creating numerous types of graphs, data types and codes for various forms of analysis

<https://stackoverflow.com/> - Stack Overflow : if an error or question is raised, Stack overflow is a site where queries can be answered. Since many people face the same problems as each other, often common issues have already been posted and resolved, allowing you to use the solutions straight away in solving your own problems.

<https://www.datacamp.com/> - Datacamp: An interactive website where you can practice solving problems using an R environment. The site courses are varied and provide you with knowledge of R code that is tailored to your own skill level.

There are many other online sites for learning to code with R, and for providing aid if problems arise, however these are the three that I most commonly use.

I hope that this code repository and list of sites can help you in your own coding efforts.