DataClean_Retail

Suprava Sahoo

30/06/2020

```r
#Following library is required for our analysis
library(tidyverse)

## Warning: package 'tidyverse' was built under R version 4.0.2

## -- Attaching packages ---------------- tidyverse 1.3.0 --

## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.1     v dplyr   1.0.0
## v tidyr   1.1.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0

## Warning: package 'ggplot2' was built under R version 4.0.2

## -- Conflicts ------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()

library(lubridate)

## Warning: package 'lubridate' was built under R version 4.0.2

##
## Attaching package: 'lubridate'

## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union

#Read the csv file
retail_dataimport <- read_csv('d:/online-retail-data-analysis-master/original
-dataset/Online_Retail.csv')

## Parsed with column specification:
## cols(
##   InvoiceNo = col_character(),
##   StockCode = col_character(),
##   Description = col_character(),
##   Quantity = col_double(),
##   InvoiceDate = col_character(),
##   UnitPrice = col_double(),
##   CustomerID = col_double(),
##   Country = col_character()
## )
```

```r
#creating a fresh copy of the data to work on so that the imported original d
ata is intact and can be reverted back easily

retail <- retail_dataimport

# What are the variables in our dataset and what are their data structure
glimpse(retail)

## Rows: 541,909
## Columns: 8
## $ InvoiceNo   <chr> "536365", "536365", "536365", "536365", "536365", "536
3...
## $ StockCode   <chr> "85123A", "71053", "84406B", "84029G", "84029E", "2275
2...
## $ Description <chr> "WHITE HANGING HEART T-LIGHT HOLDER", "WHITE METAL LAN
T...
## $ Quantity    <dbl> 6, 6, 8, 6, 6, 2, 6, 6, 6, 32, 6, 6, 8, 6, 6, 3, 2, 3,
...
## $ InvoiceDate <chr> "01-12-2010 08:26", "01-12-2010 08:26", "01-12-2010 08
:...
## $ UnitPrice   <dbl> 2.55, 3.39, 2.75, 3.39, 3.39, 7.65, 4.25, 1.85, 1.85,
1...
## $ CustomerID  <dbl> 17850, 17850, 17850, 17850, 17850, 17850, 17850, 17850
,...
## $ Country     <chr> "United Kingdom", "United Kingdom", "United Kingdom",
"...

# Check the dataset to find out which column has missing value and how many m
issing value are present corresponding to each column
retail %>%
  map(., ~sum(is.na(.)))

## $InvoiceNo
## [1] 0
##
## $StockCode
## [1] 0
##
## $Description
## [1] 1454
##
## $Quantity
## [1] 0
##
## $InvoiceDate
## [1] 0
##
## $UnitPrice
## [1] 0
##
```

```
## $CustomerID
## [1] 135080
##
## $Country
## [1] 0

#We ignore the entire row(ie observation), if any column has a missing value
retail <- retail[complete.cases(retail), ]

# Check whether all the missing values have been eliminated by summing the mi
ssing values of each column separately.
# we should  get zero for all columns

retail %>%
  map(., ~sum(is.na(.)))

## $InvoiceNo
## [1] 0
##
## $StockCode
## [1] 0
##
## $Description
## [1] 0
##
## $Quantity
## [1] 0
##
## $InvoiceDate
## [1] 0
##
## $UnitPrice
## [1] 0
##
## $CustomerID
## [1] 0
##
## $Country
## [1] 0

#Data cleaning
#Note that InvoiceDate is in <chr>, Country and Description is also in <chr>
# need to change InvoiceDate to <dttm>
# need to change TransactionID, Country and Description as factor for proper
analysis
#Idea is to replace the columns after transforming the data-type of each colu
mn, keeping their values fixed.

retail_cleaned <- retail %>%
  mutate(InvoiceDate = dmy_hm(InvoiceDate)) %>% #coerces InvoiceDate in a Dat
```

```
e Time format
  mutate(Description = factor(Description, levels = unique(Description))) %>%
  #coerces Description as a factor with each item as individual level of a fa
ctor

  mutate(Country = factor(Country, levels = unique(Country)))%>%
  mutate(InvoiceNo = factor(InvoiceNo, levels = unique(InvoiceNo))) %>%
  mutate(TotalPrice = Quantity * UnitPrice)

glimpse(retail_cleaned)

## Rows: 406,829
## Columns: 9
## $ InvoiceNo   <fct> 536365, 536365, 536365, 536365, 536365, 536365, 536365
,...
## $ StockCode   <chr> "85123A", "71053", "84406B", "84029G", "84029E", "2275
2...
## $ Description <fct> WHITE HANGING HEART T-LIGHT HOLDER, WHITE METAL LANTER
N...
## $ Quantity    <dbl> 6, 6, 8, 6, 6, 2, 6, 6, 6, 32, 6, 6, 8, 6, 6, 3, 2, 3,
...
## $ InvoiceDate <dttm> 2010-12-01 08:26:00, 2010-12-01 08:26:00, 2010-12-01
0...
## $ UnitPrice   <dbl> 2.55, 3.39, 2.75, 3.39, 3.39, 7.65, 4.25, 1.85, 1.85,
1...
## $ CustomerID  <dbl> 17850, 17850, 17850, 17850, 17850, 17850, 17850, 17850
,...
## $ Country     <fct> United Kingdom, United Kingdom, United Kingdom, United
...
## $ TotalPrice  <dbl> 15.30, 20.34, 22.00, 20.34, 20.34, 15.30, 25.50, 11.10
,...

#Save it as a RData file which we will import in the next stage
save(retail_cleaned, file = 'd:/online-retail-data-analysis-master/intermedia
te-data/retail_cleaned.RData')
```

EDA_Retail

Suprava Sahoo

30/06/2020

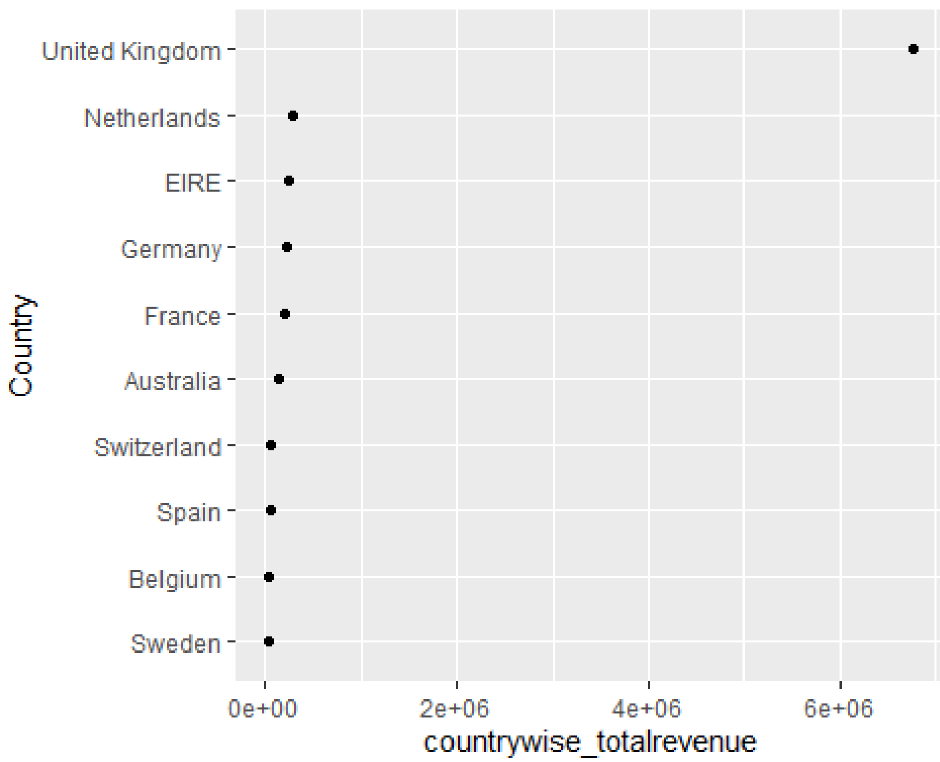
```
library(tidyverse)

## Warning: package 'tidyverse' was built under R version 4.0.2

## -- Attaching packages ---------------- tidyverse 1.3.0 --

## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.1     v dplyr   1.0.0
## v tidyr   1.1.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0

## Warning: package 'ggplot2' was built under R version 4.0.2

## -- Conflicts ------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()

library(lubridate)

## Warning: package 'lubridate' was built under R version 4.0.2

##
## Attaching package: 'lubridate'

## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union

library(forcats)

# load the cleaned retail data from step 1.Lets load the .RData file as its
load
# time is significantly faster than csv or excel files.
load('d:/online-retail-data-analysis-master/intermediate-
data/retail_cleaned.RData')
# Have a quick look at the data
glimpse(retail_cleaned)

## Rows: 406,829
## Columns: 9
## $ InvoiceNo   <fct> 536365, 536365, 536365, 536365, 536365, 536365,
536365,...
## $ StockCode   <chr> "85123A", "71053", "84406B", "84029G", "84029E",
"22752...
```

```
## $ Description <fct> WHITE HANGING HEART T-LIGHT HOLDER, WHITE METAL
LANTERN...
## $ Quantity    <dbl> 6, 6, 8, 6, 6, 2, 6, 6, 6, 32, 6, 6, 8, 6, 6, 3, 2, 3,
...
## $ InvoiceDate <dttm> 2010-12-01 08:26:00, 2010-12-01 08:26:00, 2010-12-01
0...
## $ UnitPrice   <dbl> 2.55, 3.39, 2.75, 3.39, 3.39, 7.65, 4.25, 1.85, 1.85,
1...
## $ CustomerID  <dbl> 17850, 17850, 17850, 17850, 17850, 17850, 17850,
17850,...
## $ Country     <fct> United Kingdom, United Kingdom, United Kingdom, United
...
## $ TotalPrice  <dbl> 15.30, 20.34, 22.00, 20.34, 20.34, 15.30, 25.50,
11.10,...
```

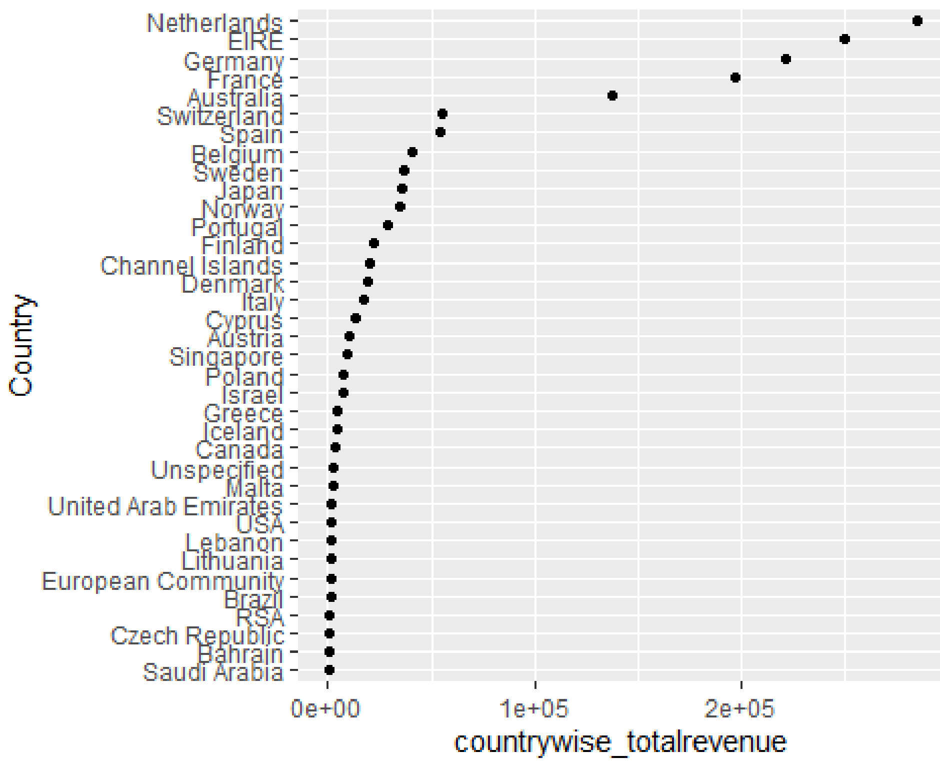
```
# How to arrange the countries based on their total sales? and which country
has
# maximum sales? Group the data countrywise then find the frequency of sales
per
# country using the count function step3 : Arrange the countries by
descending
# sales

country_mostsales <- retail_cleaned %>%
  group_by(Country) %>%
  summarize(countrywise_totalrevenue = sum(TotalPrice))%>%
  arrange(desc(countrywise_totalrevenue))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

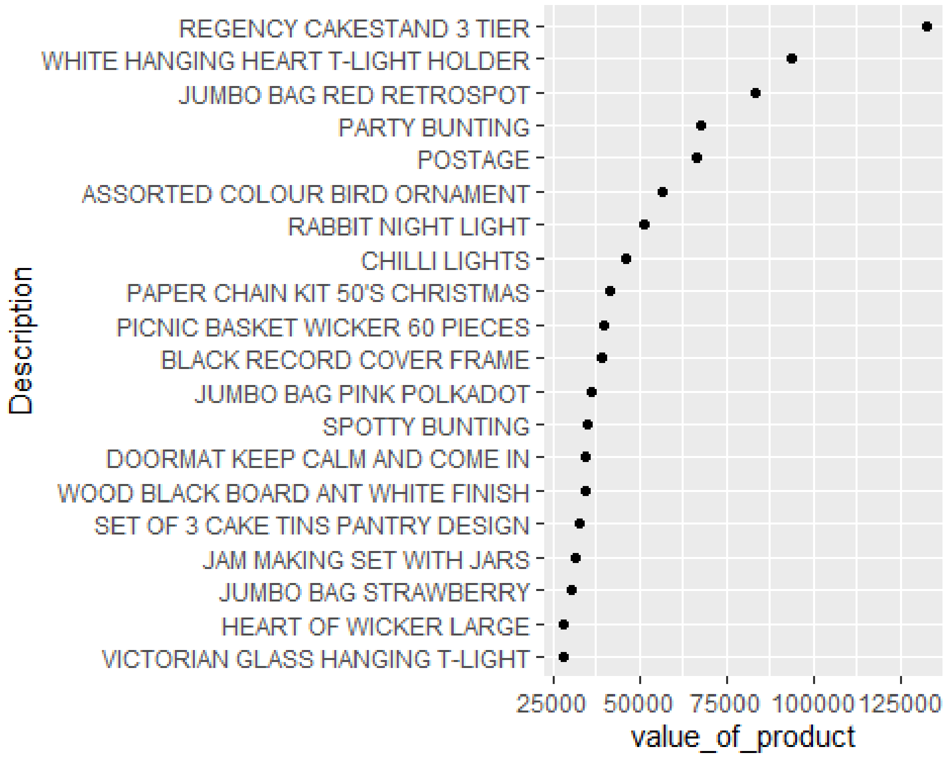
```
by_top10_countries <- country_mostsales %>%
  top_n(n = 10, wt = countrywise_totalrevenue)

# Visualize the top 10 countries as per total sales. scatterplot of country
# versus sales.Country has no natural ordering,so we use fact-reorder() to
# display the conutries as per increasing sales. In other words, The
scatterplot
# is arranged so that the country having minimum sales is plotted first and
then
# the country having second lowest sales and so on till the final country
which
# has maximum sales.

by_top10_countries %>%
  mutate(Country = fct_reorder(Country, countrywise_totalrevenue)) %>%
  ggplot(aes(Country, countrywise_totalrevenue))+
  geom_point()+
  coord_flip()
```



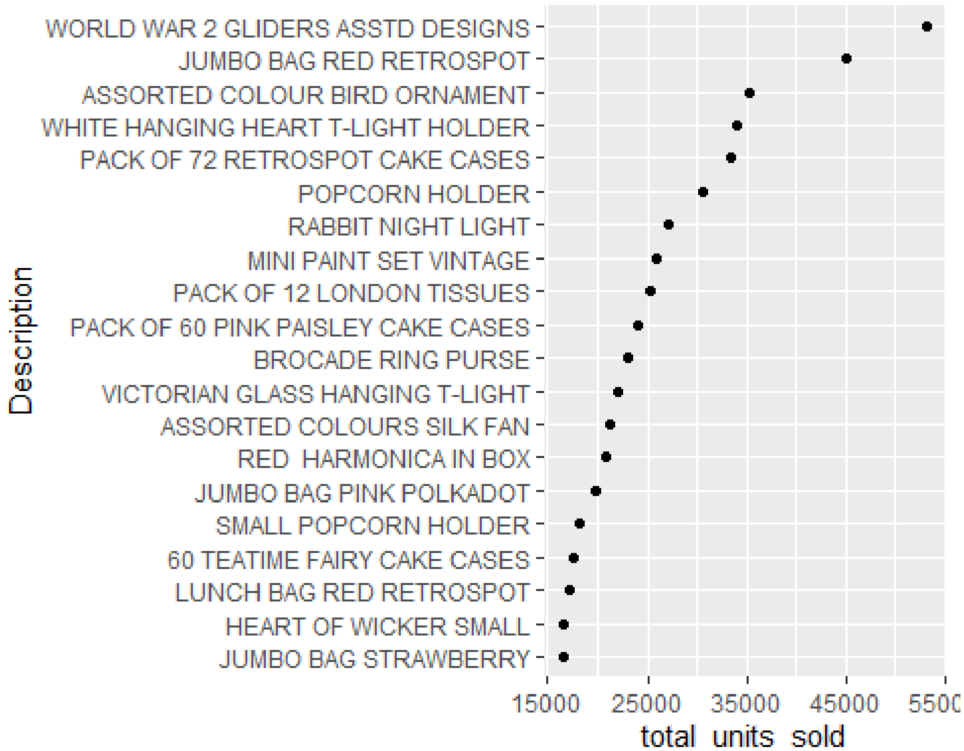

```
# So UK has the highest sale followed by Netherlands, EIRE , Germany and
France


# However, UK is the home country of the firm and that explains why this
country
# has large amount of sales. Hence lets check the trend by separating UK. We
# visualize how sales are distributed over various countries.

country_mostsales_without_uk <-  country_mostsales[-1,]

country_mostsales_without_uk %>%
  mutate(Country = fct_reorder(Country, countrywise_totalrevenue)) %>%
  ggplot(aes(Country, countrywise_totalrevenue))+
  geom_point()+
  coord_flip()
```




```
# Most valued product-The product briging largest turnover

mostvalued_product <- retail_cleaned %>%
  group_by(Description) %>%
  summarize(value_of_product = sum(TotalPrice)) %>%
  arrange(desc(value_of_product))

## `summarise()` ungrouping output (override with `.groups` argument)

# top20 most valued product

top20_valued_products <- mostvalued_product %>%
  top_n(n = 20, wt = value_of_product)

# Categorical variable like Description does not have an intrinsic order, so
we
# reorder it as per increasing count.

# graphical representation of most valued product

top20_valued_products %>%
  mutate(Description = fct_reorder(Description, value_of_product)) %>%
  ggplot(aes(Description, value_of_product))+
  geom_point()+
  coord_flip()
```

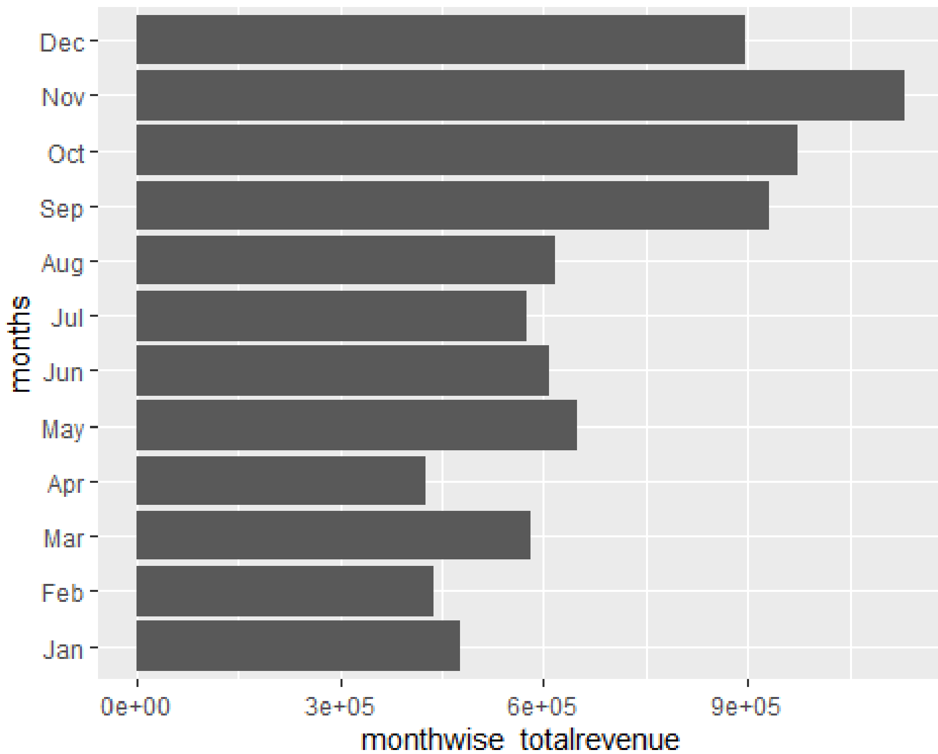



```
# So REGENCY CAKESTAND 3 TIER, WHITE HANGING HEART T-LIGHT HOLDER, JUMBO BAG
RED
# RETROSPOT are the top3 most valued products


# which product is most sold worldwide?
mostsold_product <- retail_cleaned %>%
  group_by(Description) %>%
  summarize(total_units_sold = sum(Quantity)) %>%
  arrange(desc(total_units_sold))

## `summarise()` ungrouping output (override with `.groups` argument)

# top20 most sold products--
top20_mostsoldproducts <- mostsold_product %>%
  top_n(n = 20, wt = total_units_sold)


top20_mostsoldproducts %>%
  mutate(Description = fct_reorder(Description, total_units_sold)) %>%
  ggplot(aes(Description, total_units_sold))+
  geom_point()+
  coord_flip()
```




```
# Most sold products worldwide are WORLD WAR 2 GLIDERS ASSTD DESIGNS, UMBO
BAG
# RED RETROSPOT, ASSORTED COLOUR BIRD ORNAMENT

# which customers are most valuable for the company?

by_mostvaluable_customer <- retail_cleaned %>%
  group_by(CustomerID) %>%
  summarize(customer_amount = sum(TotalPrice)) %>%
  arrange(desc(customer_amount))

## `summarise()` ungrouping output (override with `.groups` argument)

top20_mostvaluable_customer <- by_mostvaluable_customer %>%
  top_n(n = 20, wt = customer_amount)
View(top20_mostvaluable_customer)


# Which month of the year sees maximum turnover?

data_with_month <- retail_cleaned %>%
  mutate(months = month(InvoiceDate,label = T))

# the month() function separates out the month name from a date-time(dttm)
# column. We create a separate column name months with the month names of the
# sales data.
```

```r
data_with_month %>%
  group_by(months)%>%
  summarize(monthwise_totalrevenue = sum(TotalPrice))%>%
  ggplot()+
  geom_bar(mapping = aes(x = months, y = monthwise_totalrevenue), stat =
"identity")+
  coord_flip()

## `summarise()` ungrouping output (override with `.groups` argument)
```




```r
# September to December appear to be the months with highest sales.This is
not
# surprising as these months are winter months for Europian countries where
most
# sales occur and winter time is festive time.
```

Market Basket

Suprava Sahoo

30/06/2020

```
# Lets perform a marketbasket analysis to analyse which product is likely to
be sold with which product
library(arules)

## Warning: package 'arules' was built under R version 4.0.2

## Loading required package: Matrix

##
## Attaching package: 'arules'

## The following objects are masked from 'package:base':
##
##     abbreviate, write

library(arulesViz)

## Warning: package 'arulesViz' was built under R version 4.0.2

## Loading required package: grid

## Registered S3 method overwritten by 'seriation':
##   method         from
##   reorder.hclust gclus

library(tidyverse)

## Warning: package 'tidyverse' was built under R version 4.0.2

## -- Attaching packages ---------------- tidyverse 1.3.0 --

## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.1     v dplyr   1.0.0
## v tidyr   1.1.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0

## Warning: package 'ggplot2' was built under R version 4.0.2

## -- Conflicts ------------------- tidyverse_conflicts() --
## x tidyr::expand() masks Matrix::expand()
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x tidyr::pack()   masks Matrix::pack()
## x dplyr::recode() masks arules::recode()
## x tidyr::unpack() masks Matrix::unpack()
```

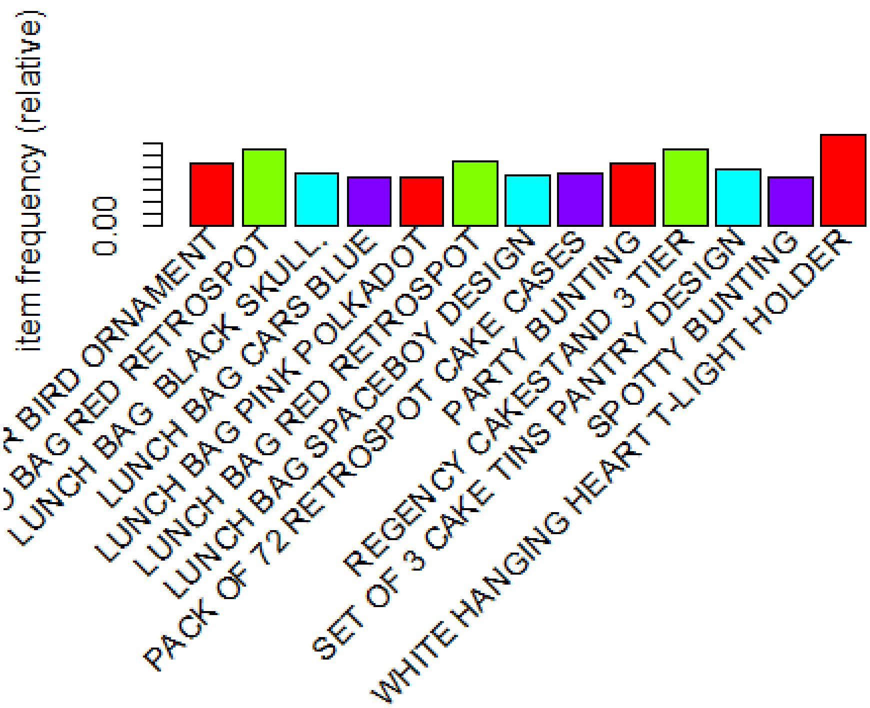
```r
library(plyr)

## --------------------------------------------------------------------------
----

## You have loaded plyr after dplyr - this is likely to cause problems.
## If you need functions from both plyr and dplyr, please load plyr first,
then dplyr:
## library(plyr); library(dplyr)

## --------------------------------------------------------------------------
----

##
## Attaching package: 'plyr'

## The following objects are masked from 'package:dplyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize

## The following object is masked from 'package:purrr':
##
##     compact

load('C:/Users/suppy/Desktop/online-retail-data-analysis-master/intermediate-
data/retail_cleaned.RData')
transaction_df <-  select(retail_cleaned, 'InvoiceNo', 'Description')

#How many unique levels of InvoiceNo and Description of the product are there
str(transaction_df)

## tibble [406,829 x 2] (S3: tbl_df/tbl/data.frame)
##  $ InvoiceNo  : Factor w/ 22190 levels "536365","536366",..: 1 1 1 1 1 1 1
2 2 3 ...
##  $ Description: Factor w/ 3885 levels "WHITE HANGING HEART T-LIGHT
HOLDER",..: 1 2 3 4 5 6 7 8 9 10 ...

#save transaction id and commodities in one file for future reference
write.csv(transaction_df,'C:/Users/suppy/Desktop/online-retail-data-analysis-
master/intermediate-data/transaction_df.csv', row.names = FALSE)

#creating a itemList from the Description column of the data.
#for each InvoiceNo,description of all the products brought together are
written together

itemList <- plyr :: ddply(transaction_df, c("InvoiceNo"),
                  function(transaction_df)paste(transaction_df$Description,
                                        collapse = ","))
#itemList
#deleting the InvoiceNO from the itemList data as this is not required
anymore
```

```
itemList$InvoiceNo <- NULL

#Write out the itemlist per transaction in a csv file
write.csv(itemList,'C:/Users/suppy/Desktop/online-retail-data-analysis-
master/intermediate-data/market_basket_tr.csv', row.names = FALSE)

#Read the csv in 'basket' format
#rm.duplicates removes duplicate items in a particular transaction.
transaction <- read.transactions('C:/Users/suppy/Desktop/online-retail-data-
analysis-master/intermediate-data/market_basket_tr.csv', format = 'basket',
quote = "", cols = NULL, sep=',', skip = 1, rm.duplicates = T)

## distribution of transactions with duplicates:
## items
##    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15
16
## 1032  529  297  212  149   96   91   62   47   48   34   23   25    8   14
12
##   17   18   19   20   22   23   24   25   26   27   28   29   30   32   33
34
##   12   10    4    3    4    4    7    2    2    2    4    3    2    1    2
1
##   36   42   44   45   49   51
##    1    1    1    1    1    1

transaction

## transactions in sparse format with
##  22190 transactions (rows) and
##  10181 items (columns)

summary(transaction)

## transactions as itemMatrix in sparse format with
##  22190 rows (elements/itemsets/transactions) and
##  10181 columns (items) and a density of 0.00177008
##
## most frequent items:
## WHITE HANGING HEART T-LIGHT HOLDER           REGENCY CAKESTAND 3 TIER
##                               1683                               1445
##            JUMBO BAG RED RETROSPOT            LUNCH BAG RED RETROSPOT
##                               1420                               1206
##                      PARTY BUNTING                            (Other)
##                               1162                             392974
##
## element (itemset/transaction) length distribution:
## sizes
##    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15
16
## 3371 1472 1044  776  761  647  616  615  637  526  559  509  487  516  548
548
```

```
##   17   18   19   20   21   22   23   24   25   26   27   28   29   30   31 
32 
##  459  435  486  417  420  339  342  303  237  269  254  205  269  236  195 
172 
##   33   34   35   36   37   38   39   40   41   42   43   44   45   46   47 
48 
##  161  171  144  118  139  104  122  121  119  111   95   88   83   91   82 
81 
##   49   50   51   52   53   54   55   56   57   58   59   60   61   62   63 
64 
##   69   72   69   57   60   70   60   53   52   48   43   38   47   37   32 
31 
##   65   66   67   68   69   70   71   72   73   74   75   76   77   78   79 
80 
##   32   33   42   34   24   27   27   19   24   35   23   23   17   18    9 
18 
##   81   82   83   84   85   86   87   88   89   90   91   92   93   94   95 
96 
##   16   19   15   19   16   11   14   12   11    6    9   18   12    9    4 
9 
##   97   98   99  100  101  102  103  104  105  106  107  108  109  110  111 
112 
##   10   10    5    7   11    5    9    6    4    2    4    6    4    2    4 
1 
##  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127 
128 
##    6    3    2   10    3    7    5    4    5    3    8    2    3    6    3 
5 
##  130  131  132  133  134  135  136  137  140  141  142  143  144  146  147 
149 
##    1    1    1    4    1    1    3    3    4    1    3    2    1    4    1 
2 
##  150  151  152  155  157  158  159  165  167  170  171  177  178  181  185 
187 
##    1    2    1    1    1    1    1    2    1    1    2    2    2    3    1 
1 
##  193  194  196  204  205  208  211  220  230  251  259  263  273  283  339 
351 
##    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1 
1 
##  356  366  379  387  422  440  442  529  533  547 
##    1    1    1    1    1    1    1    1    1    1 
## 
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##    1.00    3.00   12.00   18.02   24.00  547.00 
## 
## includes extended item information - examples:
##                       labels
## 1                          "
```

```
## 2  "10 COLOUR SPACEBOY PEN
## 3 "10 COLOUR SPACEBOY PEN"
```

```r
# Make a frequency plot of the transactions with a support of 0.05 or greater.
# This shows the the most popular gift items sold.
itemFrequencyPlot(transaction, support = .04, col = rainbow(4))
```




```r
# create association rules with a minimum support value ,where support indicates appearance of
#commodity A and B together out of total transactions of all items.
rules <- apriori(transaction, parameter = list(supp = 0.01, conf = 0.5, minlen = 2))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.5    0.1    1 none FALSE            TRUE       5    0.01      2
##  maxlen target  ext
##      10  rules TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 221
```

```
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[10181 item(s), 22190 transaction(s)] done [0.18s].
## sorting and recoding items ... [457 item(s)] done [0.01s].
## creating transaction tree ... done [0.01s].
## checking subsets of size 1 2 3 4 done [0.01s].
## writing ... [89 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
options(digits=2)
top10rules <-rules[1:10]
inspect(top10rules)
```

```
##      lhs                               rhs                             support
confidence coverage lift count
## [1]  {SUGAR}                        => {SET 3 RETROSPOT TEA}             0.010
0.95    0.011   89   226
## [2]  {SET 3 RETROSPOT TEA}          => {SUGAR}                           0.010
0.96    0.011   89   226
## [3]  {SUGAR}                        => {COFFEE}                          0.011
1.00    0.011   64   239
## [4]  {COFFEE}                       => {SUGAR}                           0.011
0.69    0.016   64   239
## [5]  {SET 3 RETROSPOT TEA}          => {COFFEE}                          0.011
1.00    0.011   64   236
## [6]  {COFFEE}                       => {SET 3 RETROSPOT TEA}             0.011
0.68    0.016   64   236
## [7]  {ALARM CLOCK BAKELIKE ORANGE} => {ALARM CLOCK BAKELIKE RED}    0.010
0.65    0.016   18   227
## [8]  {BACK DOOR}                    => {KEY FOB}                         0.010
0.99    0.010   59   229
## [9]  {KEY FOB}                      => {BACK DOOR}                       0.010
0.62    0.017   59   229
## [10] {POPPY'S PLAYHOUSE BEDROOM}   => {POPPY'S PLAYHOUSE KITCHEN}   0.011
0.79    0.014   53   237
```

```
plot(top10rules, method = "graph", engine = 'interactive')

#if A => B is the rule, confidence shows the proportion of transactions
having both A and B,
#out of total transactions having A.

#sort the rules by decreasing confidence and show top 10 rules
rules_by_confidence <- sort(rules, by ='confidence', decreasing = TRUE)
summary(rules_by_confidence)
```

```
## set of 89 rules
##
## rule length distribution (lhs + rhs):sizes
##  2  3
## 57 32
```

```
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    2.00    2.00    2.00    2.36    3.00    3.00
##
## summary of quality measures:
##     support         confidence       coverage          lift         count
##  Min.   :0.0100   Min.   :0.50   Min.   :0.010   Min.   : 8   Min.   :222
##  1st Qu.:0.0105   1st Qu.:0.56   1st Qu.:0.017   1st Qu.:13   1st Qu.:232
##  Median :0.0114   Median :0.61   Median :0.019   Median :17   Median :252
##  Mean   :0.0130   Mean   :0.64   Mean   :0.021   Mean   :25   Mean   :289
##  3rd Qu.:0.0150   3rd Qu.:0.68   3rd Qu.:0.026   3rd Qu.:23   3rd Qu.:332
##  Max.   :0.0218   Max.   :1.00   Max.   :0.040   Max.   :89   Max.   :483
##
## mining info:
##         data ntransactions support confidence
##  transaction         22190    0.01        0.5
```

```
toprules_by_confidence <- rules_by_confidence[1:10]
options(digits=2)
inspect(toprules_by_confidence)
```

```
##      lhs                                  rhs
support confidence coverage lift count
## [1]  {SUGAR}                           => {COFFEE}
0.011       1.00    0.011   64   239
## [2]  {SET 3 RETROSPOT TEA}             => {COFFEE}
0.011       1.00    0.011   64   236
## [3]  {SET 3 RETROSPOT TEA,
##       SUGAR}                           => {COFFEE}
0.010       1.00    0.010   64   226
## [4]  {SHED}                            => {KEY FOB}
0.012       0.99    0.012   59   262
## [5]  {BACK DOOR}                       => {KEY FOB}
0.010       0.99    0.010   59   229
## [6]  {SET 3 RETROSPOT TEA}             => {SUGAR}
0.010       0.96    0.011   89   226
## [7]  {COFFEE,
##       SET 3 RETROSPOT TEA}             => {SUGAR}
0.010       0.96    0.011   89   226
## [8]  {SUGAR}                           => {SET 3 RETROSPOT TEA}
0.010       0.95    0.011   89   226
## [9]  {COFFEE,
##       SUGAR}                           => {SET 3 RETROSPOT TEA}
0.010       0.95    0.011   89   226
## [10] {PINK REGENCY TEACUP AND SAUCER,
##       ROSES REGENCY TEACUP AND SAUCER} => {GREEN REGENCY TEACUP AND
SAUCER}   0.014       0.82    0.017   28   310
```

```
plot(toprules_by_confidence, method="graph",engine = 'interactive',shading =
NA)
```

```
#Lift is the factor by which, the co-occurence of A and B exceeds the
expected
 #probability of A and B co-occuring, had they been independent. So, higher
the
 #lift, higher the chance of A and B occurring together.
 # sort the rules by decreasing lift and show top 10 rules

rules_by_lift <- sort(rules, by='lift', decreasing = TRUE)
summary(rules_by_lift)

## set of 89 rules
##
## rule length distribution (lhs + rhs):sizes
##  2  3
## 57 32
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    2.00    2.00    2.00    2.36    3.00    3.00
##
## summary of quality measures:
##     support          confidence       coverage           lift         count
##  Min.   :0.0100   Min.   :0.50   Min.   :0.010   Min.   : 8   Min.   :222
##  1st Qu.:0.0105   1st Qu.:0.56   1st Qu.:0.017   1st Qu.:13   1st Qu.:232
##  Median :0.0114   Median :0.61   Median :0.019   Median :17   Median :252
##  Mean   :0.0130   Mean   :0.64   Mean   :0.021   Mean   :25   Mean   :289
##  3rd Qu.:0.0150   3rd Qu.:0.68   3rd Qu.:0.026   3rd Qu.:23   3rd Qu.:332
##  Max.   :0.0218   Max.   :1.00   Max.   :0.040   Max.   :89   Max.   :483
##
## mining info:
##         data ntransactions support confidence
##  transaction         22190    0.01        0.5

toprules_by_lift <- rules_by_lift[1:10]
options(digits=2)
inspect(toprules_by_lift)

##      lhs                                rhs                   support
confidence
## [1]  {SUGAR}                         => {SET 3 RETROSPOT TEA} 0.010   0.95
## [2]  {COFFEE,SUGAR}                  => {SET 3 RETROSPOT TEA} 0.010   0.95
## [3]  {SET 3 RETROSPOT TEA}           => {SUGAR}               0.010   0.96
## [4]  {COFFEE,SET 3 RETROSPOT TEA} => {SUGAR}                  0.010   0.96
## [5]  {SUGAR}                         => {COFFEE}              0.011   1.00
## [6]  {COFFEE}                        => {SUGAR}               0.011   0.69
## [7]  {SET 3 RETROSPOT TEA}           => {COFFEE}              0.011   1.00
## [8]  {COFFEE}                        => {SET 3 RETROSPOT TEA} 0.011   0.68
## [9]  {SET 3 RETROSPOT TEA,SUGAR}     => {COFFEE}              0.010   1.00
## [10] {SHED}                          => {KEY FOB}             0.012   0.99
##      coverage lift count
```

```
## [1]  0.011    89   226
## [2]  0.011    89   226
## [3]  0.011    89   226
## [4]  0.011    89   226
## [5]  0.011    64   239
## [6]  0.016    64   239
## [7]  0.011    64   236
## [8]  0.016    64   236
## [9]  0.010    64   226
## [10] 0.012    59   262
```

```
plot(toprules_by_lift, method="graph",engine = 'interactive',shading = NA)

#Since WHITE HANGING HEART T-LIGHT HOLDER  is the most popular item, we are
#interested in the items bought with it.
rules_lhs_white_hanging_heart_t_shirt_holder<-apriori(data=transaction,
parameter=list(supp=0.001,conf = 0.1, minlen = 2),
                                  appearance = list(default="rhs",lhs="WHITE
HANGING HEART T-LIGHT HOLDER"),
                                  control = list(verbose=F))
rules_lhs_white_hanging_heart_t_shirt_holder <-
sort(rules_lhs_white_hanging_heart_t_shirt_holder,
decreasing=TRUE,by="confidence")
inspect(rules_lhs_white_hanging_heart_t_shirt_holder)
```

```
##      lhs                                       rhs
support confidence coverage lift count
## [1]  {WHITE HANGING HEART T-LIGHT HOLDER} => {RED HANGING HEART T-LIGHT
HOLDER}   0.0167       0.22    0.076  8.0   371
## [2]  {WHITE HANGING HEART T-LIGHT HOLDER} => {WOODEN PICTURE FRAME WHITE
FINISH}  0.0111       0.15    0.076  4.0   246
## [3]  {WHITE HANGING HEART T-LIGHT HOLDER} => {HEART OF WICKER LARGE}
0.0108       0.14    0.076  4.4   239
## [4]  {WHITE HANGING HEART T-LIGHT HOLDER} => {HEART OF WICKER SMALL}
0.0104       0.14    0.076  3.6   230
## [5]  {WHITE HANGING HEART T-LIGHT HOLDER} => {NATURAL SLATE HEART
CHALKBOARD}     0.0103       0.14    0.076  3.5   229
## [6]  {WHITE HANGING HEART T-LIGHT HOLDER} => {PARTY BUNTING}
0.0101       0.13    0.076  2.5   224
## [7]  {WHITE HANGING HEART T-LIGHT HOLDER} => {CANDLEHOLDER PINK HANGING
HEART}   0.0099       0.13    0.076  8.8   219
## [8]  {WHITE HANGING HEART T-LIGHT HOLDER} => {ASSORTED COLOUR BIRD
ORNAMENT}      0.0097       0.13    0.076  2.5   215
## [9]  {WHITE HANGING HEART T-LIGHT HOLDER} => {JUMBO BAG RED RETROSPOT}
0.0096       0.13    0.076  2.0   213
## [10] {WHITE HANGING HEART T-LIGHT HOLDER} => {WOODEN FRAME ANTIQUE WHITE}
0.0092       0.12    0.076  3.6   204
## [11] {WHITE HANGING HEART T-LIGHT HOLDER} => {REGENCY CAKESTAND 3 TIER}
0.0091       0.12    0.076  1.8   201
## [12] {WHITE HANGING HEART T-LIGHT HOLDER} => {LUNCH BAG RED RETROSPOT}
```

```
0.0088       0.12    0.076  2.1   196
## [13] {WHITE HANGING HEART T-LIGHT HOLDER} => {LUNCH BAG PINK POLKADOT}
0.0079       0.10    0.076  2.6   175
## [14] {WHITE HANGING HEART T-LIGHT HOLDER} => {LUNCH BAG  BLACK SKULL.}
0.0076       0.10    0.076  2.2   169

gifts_with_tshirtholder <- rules_lhs_white_hanging_heart_t_shirt_holder[1:10]

plot(gifts_with_tshirtholder, method="graph",engine = 'interactive',shading =
NA)
```