Amazon_final.R

chidam

Sat Apr 23 15:21:28 2016

```
setwd('/Users/chidam/Desktop/Official Course documents/Semester 2/MKT591/Amazon/')

#Loading Amazon Reviews DataSet
Amzreviews = read.csv("Amazon_reviews.csv")
str(Amzreviews)
```

```
## 'data.frame':    31690 obs. of  20 variables:
##  $ review_id                : int  8 21 9763 20854 20856 20857 14452 20901 20855 2
...
##  $ prod_id                  : int  1 1 255 661 661 661 494 662 661 1 ...
##  $ review                   : Factor w/ 19195 levels "","- Even without using data
/internet, the battery gets empty in about half a day.', , - The phone is generally e
xtremely slow.', "| __truncated__,..: 9594 3916 4950 2612 673 197 18802 3756 10832 96
80 ...
##  $ rev_heading              : Factor w/ 14715 levels "-_- what now??",..: 2006 134
33 6612 4462 13 4462 1524 4520 4462 148 ...
##  $ rev_date                 : Factor w/ 2254 levels "2004-08-01","2004-09-21",..:
2118 2106 2164 2182 1550 1885 1322 1960 2192 2104 ...
##  $ rev_user                 : Factor w/ 16916 levels "- MerkuriuX -",..: 14997 147
1 13194 4452 4993 473 10398 5613 5231 15354 ...
##  $ rev_rating               : num  5 5 5 5 5 5 3 4 5 3 ...
##  $ rev_user_ranking         : logi  NA NA NA NA NA NA ...
##  $ rev_url                  : Factor w/ 31287 levels "/gp/customer-reviews/R100OBP
6E2AG9E?ASIN=B0029ZA2W0",..: 24364 27919 26881 1611 1076 12299 18709 21829 11326 1869
4 ...
##  $ polarity                 : num  0.081 0.265 -0.406 0.342 1.325 ...
##  $ aspects                  : logi  NA NA NA NA NA NA ...
##  $ concepts                 : logi  NA NA NA NA NA NA ...
##  $ pleasantness             : num  0.002 0.259 0.312 0.455 1.429 ...
##  $ sensitivity              : num  0.061 0.003 -0.231 -0.045 -0.034 0 0.126 0 0.15
4 0.004 ...
##  $ attention                : num  -0.08 0.059 0.5 0.054 1.318 ...
##  $ aptitude                 : num  0.04 0.307 0.601 0.546 1.173 ...
##  $ sentistrengthsatisfaction: Factor w/ 6 levels "negative","neutral",..: 6 6 6 3
6 2 6 6 4 6 ...
##  $ sentistrengthpolarity    : num  8.58 10.3 1.53 0.46 4.18 ...
##  $ stanfordnlppolarity      : num  0.36 0.674 0.147 0.952 1 ...
##  $ stanfordnlpsatisfaction  : int  2 4 1 5 5 3 2 5 5 2 ...
```

```
Amzproducts = read.csv("Amazon_Products.csv")
str(Amzproducts)
```

```
## 'data.frame':    1127 obs. of  8 variables:
##  $ prod_id    : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ prod_brand : Factor w/ 9 levels "Apple","Asus",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ prod_name  : Factor w/ 1108 levels "((Unlocked))BlackBerry Bold 9650 Unlocked C
lean ESN Smartphone non camera",..: 331 326 328 339 330 329 911 327 334 1108 ...
##  $ prod_desc  : Factor w/ 1032 levels "- - - - SPECIFICATIONS - - - - 2G Network G
SM 850 / 900 / 1800 / 1900 3G Network HSDPA 850 / 900 / 1900 / 2100 Announced 2011, "
| __truncated__,..: 219 270 265 207 269 266 82 309 593 86 ...
##  $ prod_cost  : num  199 299 299 229 299 ...
##  $ prod_rating: num  4.2 4.2 4.2 4.2 4.2 4.2 4 4 4.7 2 ...
##  $ rev_count  : int  1726 1726 1726 1726 1726 1726 53 62 48 5 ...
##  $ prod_url   : Factor w/ 1117 levels "http://www.amazon.com/1320-RM-995-Smartphon
e-Unlocked-Yellow/dp/B00KCV1AYC",..: 1114 1113 1112 353 352 351 896 350 1117 1107 ...
```

```
#Selecting Only the required columns
reviews = Amzreviews[,c(1,2,3,4,5,6,7)]
str(reviews)
```

```
## 'data.frame':    31690 obs. of  7 variables:
##  $ review_id  : int  8 21 9763 20854 20856 20857 14452 20901 20855 2 ...
##  $ prod_id    : int  1 1 255 661 661 661 494 662 661 1 ...
##  $ review     : Factor w/ 19195 levels "","- Even without using data/internet, the
battery gets empty in about half a day.', , - The phone is generally extremely slow.'
, "| __truncated__,..: 9594 3916 4950 2612 673 197 18802 3756 10832 9680 ...
##  $ rev_heading: Factor w/ 14715 levels "-_- what now??",..: 2006 13433 6612 4462 1
3 4462 1524 4520 4462 148 ...
##  $ rev_date   : Factor w/ 2254 levels "2004-08-01","2004-09-21",..: 2118 2106 2164
2182 1550 1885 1322 1960 2192 2104 ...
##  $ rev_user   : Factor w/ 16916 levels "- MerkuriuX -",..: 14997 1471 13194 4452 4
993 473 10398 5613 5231 15354 ...
##  $ rev_rating : num  5 5 5 5 5 5 3 4 5 3 ...
```

```
#Merging the review heading and the review
reviews$review=paste(reviews$rev_heading, reviews$review, sep = ",")

#Converting ratings into classes
#reviews$rev_rating[reviews$rev_rating == 5] = 5
#reviews$rev_rating[reviews$rev_rating == 4] = 5
#reviews$rev_rating[reviews$rev_rating == 3] = 3
#reviews$rev_rating[reviews$rev_rating == 2] = 1
#reviews$rev_rating[reviews$rev_rating == 1] = 1
reviews$rev_rating = as.factor(reviews$rev_rating)

#Text mining & Building Corpus
library(tm)
```

```
## Loading required package: NLP
```

```
## Warning: package 'NLP' was built under R version 3.2.3
```

```
library(quanteda)
```

```
## Warning: package 'quanteda' was built under R version 3.2.3
```

```
##
## Attaching package: 'quanteda'
##
## The following objects are masked from 'package:tm':
##
##     as.DocumentTermMatrix, stopwords
##
## The following object is masked from 'package:NLP':
##
##     ngrams
##
## The following object is masked from 'package:stats':
##
##     df
##
## The following object is masked from 'package:base':
##
##     sample
```

```
myCorpus = Corpus(VectorSource(reviews$review))
myCorpus = corpus(myCorpus)
mydfm = dfm(myCorpus, verbose = TRUE, toLower = TRUE, removeNumbers = TRUE,
            removePunct = TRUE, removeSeparators = TRUE,
            stem = TRUE, ignoredFeatures = c(stopwords("english")), ngrams = 1:3)
```

```
## Creating a dfm from a corpus ...
##    ... lowercasing
##    ... tokenizing
##    ... indexing documents: 31,690 documents
##    ... indexing features: 2,010,959 feature types
##    ... removed 1,483,244 features, from 174 supplied (glob) feature types
##    ... stemming features (English), trimmed 29657 feature variants
##    ... created a 31690 x 498058 sparse dfm
##    ... complete.
## Elapsed time: 151.324 seconds.
```

```
dtm = as.DocumentTermMatrix(mydfm, weighting = weightTfIdf)
```

```
## Warning in convert.dfm(x, to = "tm", ...): Argument weighting not used.
```

```
dtm2 = removeSparseTerms(dtm, 0.97)
```

```
#Splitting the data into train & test
library(caTools)
library(e1071)
Label = as.data.frame(as.matrix(dtm2))
Label$rev_rating = reviews$rev_rating
Label$prod_id = reviews$prod_id
Label$review_id = reviews$review_id
set.seed(100)

#sample = sample.split(Label$rev_rating, SplitRatio = 0.75)
sample = sample.split(Label, SplitRatio = 0.75)
train = subset(Label, sample == TRUE)
test = subset(Label, sample == FALSE)

#NaiveBayes Model
model = naiveBayes(rev_rating ~ .-(prod_id+review_id), data = train, laplace = 3)
summary(model)
```

```
##         Length Class  Mode
## apriori   5    table  numeric
## tables  422    -none- list
## levels    5    -none- character
## call      4    -none- call
```

```
pred = predict(model,test)
table(test$rev_rating, pred)
```

```
##    pred
##        1    2    3    4    5
##   1  976  261  100   65  144
##   2  139  169   57   34   46
##   3  114   97  131   80  124
##   4  132  152  152  284  449
##   5  312  334  281  577 2730
```

```
sum(diag(table(test$rev_rating, pred)))
```

```
## [1] 4290
```

```
pred1 = predict(model, Label)

#Adding predictions
Amzreviews$prediction = pred1

#Joining reviews and products datasets
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
##
## The following objects are masked from 'package:stats':
##
##     filter, lag
##
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(sqldf)
```

```
## Loading required package: gsubfn
## Loading required package: proto
```

```
## Warning in doTryCatch(return(expr), name, parentenv, handler): unable to load shar
ed object '/Library/Frameworks/R.framework/Resources/modules//R_X11.so':
##   dlopen(/Library/Frameworks/R.framework/Resources/modules//R_X11.so, 6): Library
not loaded: /opt/X11/lib/libSM.6.dylib
##   Referenced from: /Library/Frameworks/R.framework/Resources/modules//R_X11.so
##   Reason: image not found
```

```
## Could not load tcltk.  Will use slower R code instead.
## Loading required package: RSQLite
## Loading required package: DBI
```

```
library(gsubfn)
library(proto)
library(RSQLite)
library(DBI)
cond = 'select * from Amzproducts left outer join Amzreviews on Amzproducts.prod_id =
Amzreviews.prod_id'
Amzcons = sqldf(cond)

unique(Amzcons$prod_brand)
```

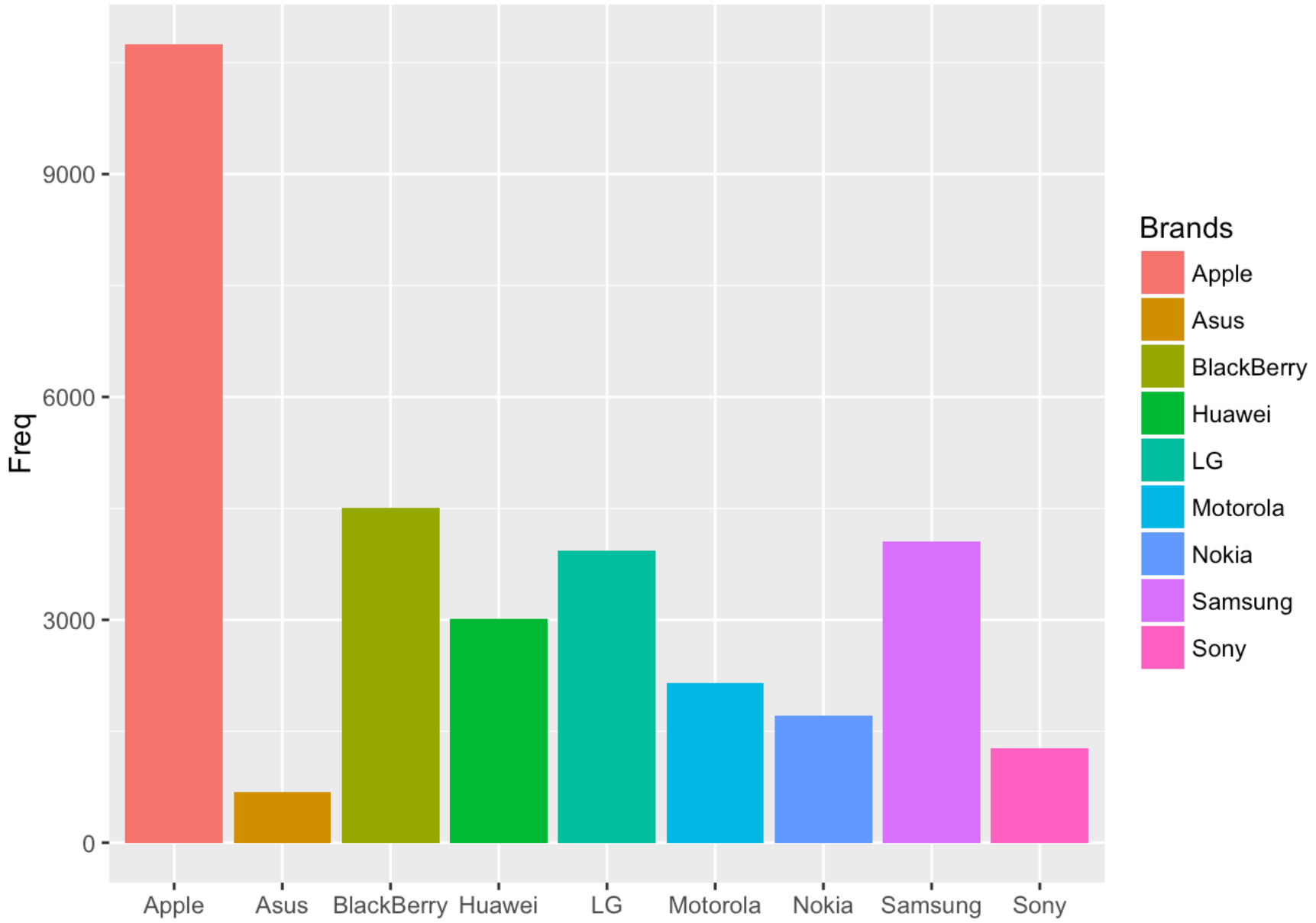
```
## [1] Asus       Apple      Huawei     BlackBerry Nokia      LG
## [7] Motorola   Samsung    Sony
## Levels: Apple Asus BlackBerry Huawei LG Motorola Nokia Samsung Sony
```

```
## Brand Traction in the Market:
brands<-data.frame(table(Amzcons$prod_brand))
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.2.4
```

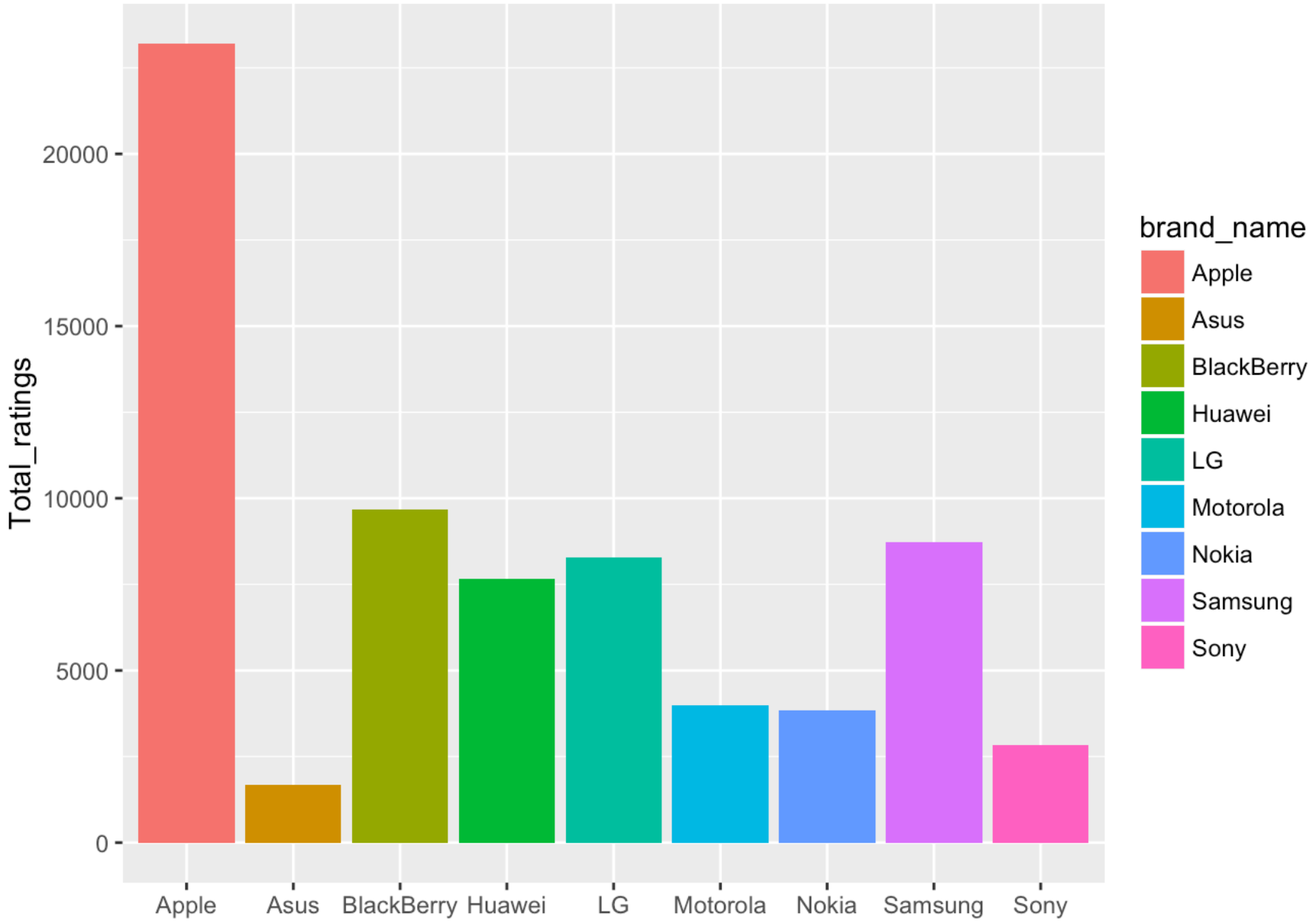
```
##
## Attaching package: 'ggplot2'
##
## The following object is masked from 'package:NLP':
##
##     annotate
```

```
names(brands) <- c('Brands','Freq')
ggplot(data=brands, aes(x=Brands, y=Freq, fill=Brands)) +
geom_bar(stat="identity", position="dodge")
```

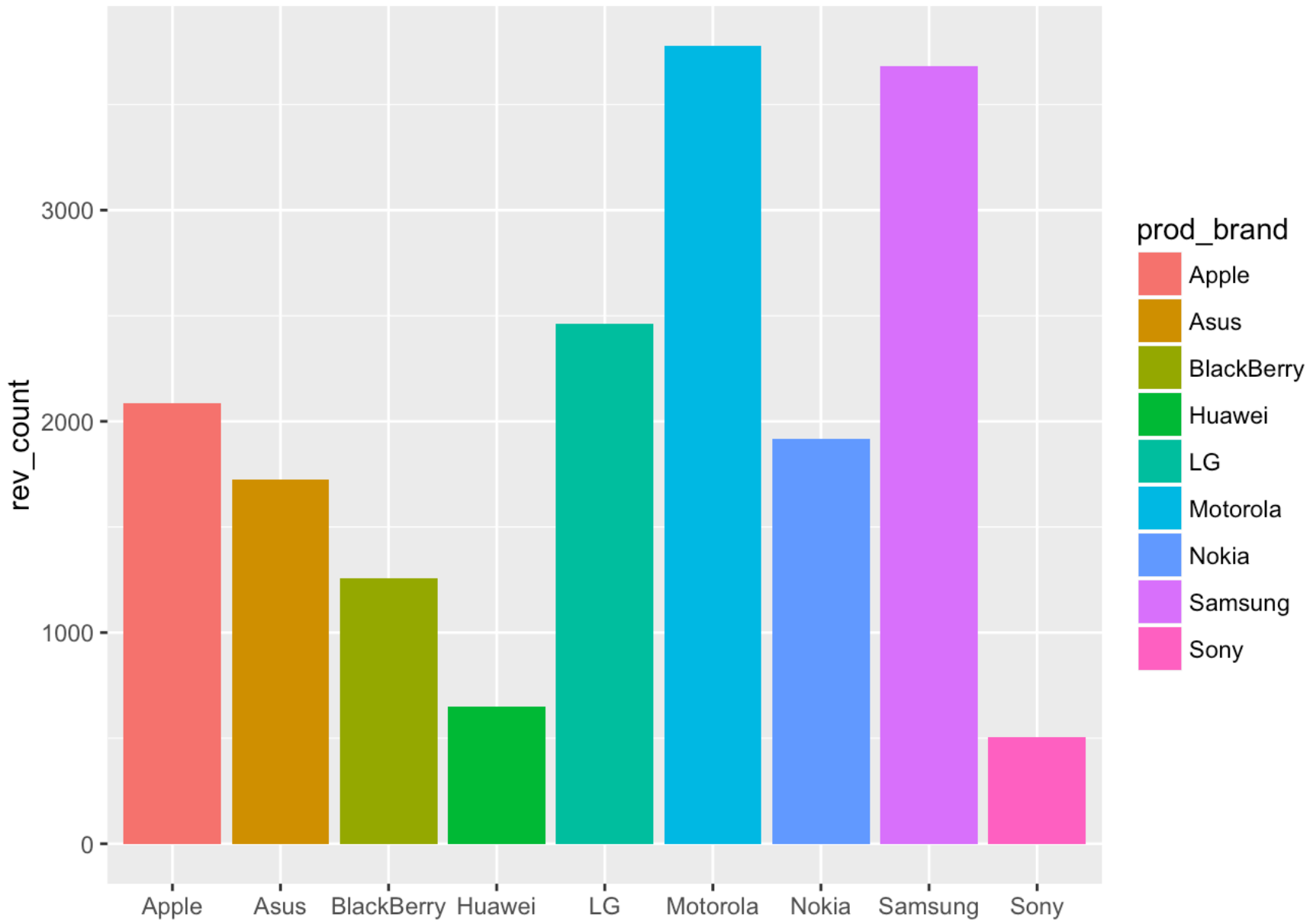



```
## Brand wise sentiments stats
brand_pred <- data.frame(table(Amzcons$prod_brand, Amzcons$prediction))
names(brand_pred) <-c('brand_name','rating','counts')
brand_pred<-sqldf('select brand_name,counts*rating as Total_ratings from brand_pred
      group by brand_name')

ggplot(data=brand_pred, aes(x=brand_name, y=Total_ratings, fill=brand_name)) +
  geom_bar(stat="identity", position="dodge")
```




```
## Most pouplar phone in each brand
most_pop_in_each_brand <- sqldf('select A.prod_brand, A.prod_name, A.rev_count
      from Amzproducts A join
      (select max(B.rev_count) as max_count, B.prod_name
      from Amzproducts B
      group by B.prod_brand) T
      on A.rev_count = T.max_count and
      A.prod_name = T.prod_name')
ggplot(data=most_pop_in_each_brand, aes(x=prod_brand, y=rev_count, fill=prod_brand))
+
  geom_bar(stat="identity", position="dodge")
```




```
#WORDCLOUD - APPLE

apple = Amzcons[Amzcons$prod_brand == "Apple",]

#Converting the text into vector source and setting up corpus
myCorpus = Corpus(VectorSource(apple$review))
myCorpus = corpus(myCorpus)
mydfm = dfm(myCorpus, verbose = TRUE, toLower = TRUE, removeNumbers = TRUE,
            removePunct = TRUE, removeSeparators = TRUE,
            stem = FALSE, ignoredFeatures = c(stopwords("english")))
```

```
## Creating a dfm from a corpus ...
##    ... lowercasing
##    ... tokenizing
##    ... indexing documents: 10,749 documents
##    ... indexing features: 10,574 feature types
##    ... removed 161 features, from 174 supplied (glob) feature types
##    ... created a 10749 x 10413 sparse dfm
##    ... complete.
## Elapsed time: 1.504 seconds.
```

```
dtm = as.DocumentTermMatrix(mydfm, weighting = weightTfIdf)
```

```
## Warning in convert.dfm(x, to = "tm", ...): Argument weighting not used.
```

```
## Warning in weighting(x): empty document(s): 2946 3047 5113 5114 5397 5548
## 5649 5700 5701 5752 6048 6710 7011 7325 7426 7598 7599 7600 7693 7936 7937
## 8055 8148 8320 8542 8593 8631 8632 8683 8882 8922 8975 9068 9119 9233 9249
## 9339 9654 9655 9806 9807 9891 9892 9893 9894 9895 9946 9947 9948 9949 9950
## 9951 9952 10003 10004 10005 10006 10038 10039 10132 10133 10134 10185 10186
## 10237 10238 10239 10240 10241 10292 10425 10426 10427 10466 10467 10468
## 10486 10487 10488 10489 10490 10562 10563 10564 10565 10566 10567 10589
## 10640 10641 10642 10643 10694 10695 10696 10697 10698 10699 10700 10701
## 10702 10703 10704 10705 10706 10707 10708 10709 10710 10711 10712 10713
## 10714 10715 10716 10717 10718 10719 10720 10721 10722 10723 10724 10725
## 10726 10727 10728 10729 10730 10731 10732 10733 10734 10735 10736 10737
## 10738 10739 10740 10741 10742 10743 10744 10745 10746 10747 10748 10749
```

```
dtm2 = removeSparseTerms(dtm, 0.97)
dtm2 = as.matrix(dtm2)

#Finding the Frequency
freq = colSums(dtm2)
freq = sort(freq, decreasing = TRUE)


library(wordcloud)
```

```
## Loading required package: RColorBrewer
```

```
words = names(freq)

wordcloud(words[1:30], freq[1:30], random.color = FALSE, colors = "BLUE")
```


```
#WORDCLOUD - ASUS

asus = Amzcons[Amzcons$prod_brand == "Asus",]

#Converting the text into vector source and setting up corpus
myCorpus = Corpus(VectorSource(asus$review))
myCorpus = corpus(myCorpus)
mydfm = dfm(myCorpus, verbose = TRUE, toLower = TRUE, removeNumbers = TRUE,
            removePunct = TRUE, removeSeparators = TRUE,
            stem = FALSE, ignoredFeatures = c(stopwords("english")))
```

```
## Creating a dfm from a corpus ...
##    ... lowercasing
##    ... tokenizing
##    ... indexing documents: 675 documents
##    ... indexing features: 4,398 feature types
##    ... removed 150 features, from 174 supplied (glob) feature types
##    ... created a 675 x 4248 sparse dfm
##    ... complete.
## Elapsed time: 0.242 seconds.
```

```
dtm = as.DocumentTermMatrix(mydfm, weighting = weightTfIdf)
```

```
## Warning in convert.dfm(x, to = "tm", ...): Argument weighting not used.
```

```
## Warning in weighting(x): empty document(s): 449 500 549 550 651 652
```

```
dtm2 = removeSparseTerms(dtm, 0.97)
dtm2 = as.matrix(dtm2)

#Finding the Frequency
freq = colSums(dtm2)
freq = sort(freq, decreasing = TRUE)



words = names(freq)

wordcloud(words[1:30], freq[1:30], random.color = FALSE, colors = "BLUE")
```

```
#WORDCLOUD - HuaWei

Huawei = Amzcons[Amzcons$prod_brand == "Huawei",]
#Converting the text into vector source and setting up corpus
myCorpus = Corpus(VectorSource(Huawei$review))
myCorpus = corpus(myCorpus)
mydfm = dfm(myCorpus, verbose = TRUE, toLower = TRUE, removeNumbers = TRUE,
            removePunct = TRUE, removeSeparators = TRUE,
            stem = FALSE, ignoredFeatures = c(stopwords("english")))
```

```
## Creating a dfm from a corpus ...
##    ... lowercasing
##    ... tokenizing
##    ... indexing documents: 3,018 documents
##    ... indexing features: 8,110 feature types
##    ... removed 154 features, from 174 supplied (glob) feature types
##    ... created a 3018 x 7956 sparse dfm
##    ... complete.
## Elapsed time: 0.319 seconds.
```

```
dtm = as.DocumentTermMatrix(mydfm, weighting = weightTfIdf)
```

```
## Warning in convert.dfm(x, to = "tm", ...): Argument weighting not used.
```

```
## Warning in weighting(x): empty document(s): 1 944 1617 1671 1699 1760 1937
## 1977 2053 2179 2180 2181 2182 2183 2184 2275 2276 2277 2328 2329 2330 2331
## 2332 2333 2334 2593 2610 2621 2637 2690 2741 2742 2743 2744 2745 2746 2747
## 2748 2749 2771 2838 2839 2840 2841 2866 2883 2884 2885 2886 2887 2906 2907
## 2908 2938 2994 2995 2996 2997 2998
```

```
dtm2 = removeSparseTerms(dtm, 0.97)
dtm2 = as.matrix(dtm2)

#Finding the Frequency
freq = colSums(dtm2)
freq = sort(freq, decreasing = TRUE)


words = names(freq)

wordcloud(words[1:30], freq[1:30], random.color = FALSE, colors = "BLUE")
```


```
#WORDCLOUD - NOKIA

Nokia = Amzcons[Amzcons$prod_brand == "Nokia",]
#Converting the text into vector source and setting up corpus
myCorpus = Corpus(VectorSource(Nokia$review))
myCorpus = corpus(myCorpus)
mydfm = dfm(myCorpus, verbose = TRUE, toLower = TRUE, removeNumbers = TRUE,
            removePunct = TRUE, removeSeparators = TRUE,
            stem = FALSE, ignoredFeatures = c(stopwords("english")))
```

```
## Creating a dfm from a corpus ...
##    ... lowercasing
##    ... tokenizing
##    ... indexing documents: 1,710 documents
##    ... indexing features: 12,536 feature types
##    ... removed 158 features, from 174 supplied (glob) feature types
##    ... created a 1710 x 12378 sparse dfm
##    ... complete.
## Elapsed time: 0.492 seconds.
```

```
dtm = as.DocumentTermMatrix(mydfm, weighting = weightTf)
```

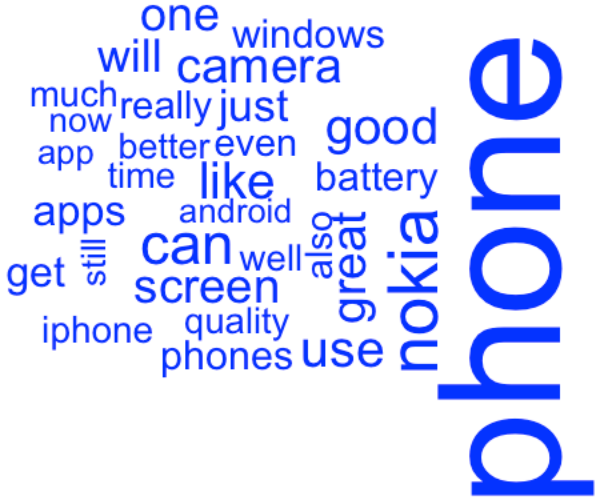
```
## Warning in convert.dfm(x, to = "tm", ...): Argument weighting not used.
```

```
dtm2 = removeSparseTerms(dtm, 0.97)
dtm2 = as.matrix(dtm2)

#Finding the Frequency
freq = colSums(dtm2)
freq = sort(freq, decreasing = TRUE)


words = names(freq)

wordcloud(words[1:30], freq[1:30], random.color = FALSE, colors = "BLUE")
```



```
#WORDCLOUD - BlackBerry

Blackberry = Amzcons[Amzcons$prod_brand == "BlackBerry",]

#Converting the text into vector source and setting up corpus
myCorpus = Corpus(VectorSource(Blackberry$review))
myCorpus = corpus(myCorpus)
mydfm = dfm(myCorpus, verbose = TRUE, toLower = TRUE, removeNumbers = TRUE,
            removePunct = TRUE, removeSeparators = TRUE,
            stem = FALSE, ignoredFeatures = c(stopwords("english")))
```

```
## Creating a dfm from a corpus ...
##    ... lowercasing
##    ... tokenizing
##    ... indexing documents: 4,511 documents
##    ... indexing features: 15,253 feature types
##    ... removed 161 features, from 174 supplied (glob) feature types
##    ... created a 4511 x 15092 sparse dfm
##    ... complete.
## Elapsed time: 0.757 seconds.
```

```
dtm = as.DocumentTermMatrix(mydfm, weighting = weightTfIdf)
```

```
## Warning in convert.dfm(x, to = "tm", ...): Argument weighting not used.
```

```
## Warning in weighting(x): empty document(s): 548 1037 1038 1139 1361 1512
## 1663 1843 2073 2124 2779 2949 3020 3400 3608 3609 3666 3771 3871 3968 3969
## 3970 3971 3972 4065 4066 4167 4168 4169 4442
```

```
dtm2 = removeSparseTerms(dtm, 0.97)
dtm2 = as.matrix(dtm2)

#Finding the Frequency
freq = colSums(dtm2)
freq = sort(freq, decreasing = TRUE)


words = names(freq)

wordcloud(words[1:30], freq[1:30], random.color = FALSE, colors = "BLUE")
```


```
#WORDCLOUD - LG

LG = Amzcons[Amzcons$prod_brand == "LG",]

#Converting the text into vector source and setting up corpus
myCorpus = Corpus(VectorSource(LG$review))
myCorpus = corpus(myCorpus)
mydfm = dfm(myCorpus, verbose = TRUE, toLower = TRUE, removeNumbers = TRUE,
            removePunct = TRUE, removeSeparators = TRUE,
            stem = FALSE, ignoredFeatures = c(stopwords("english")))
```

```
## Creating a dfm from a corpus ...
##    ... lowercasing
##    ... tokenizing
##    ... indexing documents: 3,935 documents
##    ... indexing features: 13,029 feature types
##    ... removed 161 features, from 174 supplied (glob) feature types
##    ... created a 3935 x 12868 sparse dfm
##    ... complete.
## Elapsed time: 1.014 seconds.
```

```
dtm = as.DocumentTermMatrix(mydfm, weighting = weightTfIdf)
```

```
## Warning in convert.dfm(x, to = "tm", ...): Argument weighting not used.
```

```
## Warning in weighting(x): empty document(s): 919 1968 3046 3177 3845
```

```
dtm2 = removeSparseTerms(dtm, 0.97)
dtm2 = as.matrix(dtm2)

#Finding the Frequency
freq = colSums(dtm2)
freq = sort(freq, decreasing = TRUE)


words = names(freq)

wordcloud(words[1:30], freq[1:30], random.color = FALSE, colors = "BLUE")
```

```
#WORDCLOUD - Motorola

mtr = Amzcons[Amzcons$prod_brand == "Motorola",]

#Converting the text into vector source and setting up corpus
myCorpus = Corpus(VectorSource(mtr$review))
myCorpus = corpus(myCorpus)
mydfm = dfm(myCorpus, verbose = TRUE, toLower = TRUE, removeNumbers = TRUE,
            removePunct = TRUE, removeSeparators = TRUE,
            stem = FALSE, ignoredFeatures = c(stopwords("english")))
```

```
## Creating a dfm from a corpus ...
##    ... lowercasing
##    ... tokenizing
##    ... indexing documents: 2,147 documents
##    ... indexing features: 11,829 feature types
##    ... removed 158 features, from 174 supplied (glob) feature types
##    ... created a 2147 x 11671 sparse dfm
##    ... complete.
## Elapsed time: 0.532 seconds.
```

```
dtm = as.DocumentTermMatrix(mydfm, weighting = weightTfIdf)
```

```
## Warning in convert.dfm(x, to = "tm", ...): Argument weighting not used.
```

```
## Warning in weighting(x): empty document(s): 1390 1441 1542 1739 1770 1771
## 1831 1832 1994 2059 2060 2061 2142 2143 2144 2145 2146 2147
```

```
dtm2 = removeSparseTerms(dtm, 0.97)
dtm2 = as.matrix(dtm2)
#Finding the Frequency
freq = colSums(dtm2)
freq = sort(freq, decreasing = TRUE)

words = names(freq)

wordcloud(words[1:30], freq[1:30], random.color = FALSE, colors = "BLUE")
```


```
#WORDCLOUD - Samsung

Samsung = Amzcons[Amzcons$prod_brand == "Samsung",]
#Converting the text into vector source and setting up corpus
myCorpus = Corpus(VectorSource(Samsung$review))
myCorpus = corpus(myCorpus)
mydfm = dfm(myCorpus, verbose = TRUE, toLower = TRUE, removeNumbers = TRUE,
            removePunct = TRUE, removeSeparators = TRUE,
            stem = FALSE, ignoredFeatures = c(stopwords("english")))
```

```
## Creating a dfm from a corpus ...
##    ... lowercasing
##    ... tokenizing
##    ... indexing documents: 4,060 documents
##    ... indexing features: 14,277 feature types
##    ... removed 163 features, from 174 supplied (glob) feature types
##    ... created a 4060 x 14114 sparse dfm
##    ... complete.
## Elapsed time: 1.089 seconds.
```

```
dtm = as.DocumentTermMatrix(mydfm, weighting = weightTfIdf)
```

```
## Warning in convert.dfm(x, to = "tm", ...): Argument weighting not used.
```

```
## Warning in weighting(x): empty document(s): 258 928 1128 2365 2515 3203
## 3404 3405 3444 3445 3496 3774 4007
```

```
dtm2 = removeSparseTerms(dtm, 0.97)
dtm2 = as.matrix(dtm2)
#Finding the Frequency
freq = colSums(dtm2)
freq = sort(freq, decreasing = TRUE)

words = names(freq)

wordcloud(words[1:30], freq[1:30], random.color = FALSE, colors = "BLUE")
```

```
## Warning in wordcloud(words[1:30], freq[1:30], random.color = FALSE, colors
## = "BLUE"): works could not be fit on page. It will not be plotted.
```

```
#WORDCLOUD - Sony

Sony = Amzcons[Amzcons$prod_brand == "Sony",]
#Converting the text into vector source and setting up corpus
myCorpus = Corpus(VectorSource(Sony$review))
myCorpus = corpus(myCorpus)
mydfm = dfm(myCorpus, verbose = TRUE, toLower = TRUE, removeNumbers = TRUE,
            removePunct = TRUE, removeSeparators = TRUE,
            stem = FALSE, ignoredFeatures = c(stopwords("english")))
```

```
## Creating a dfm from a corpus ...
##    ... lowercasing
##    ... tokenizing
##    ... indexing documents: 1,273 documents
##    ... indexing features: 8,089 feature types
##    ... removed 151 features, from 174 supplied (glob) feature types
##    ... created a 1273 x 7938 sparse dfm
##    ... complete.
## Elapsed time: 0.231 seconds.
```

```
dtm = as.DocumentTermMatrix(mydfm, weighting = weightTfIdf)
```

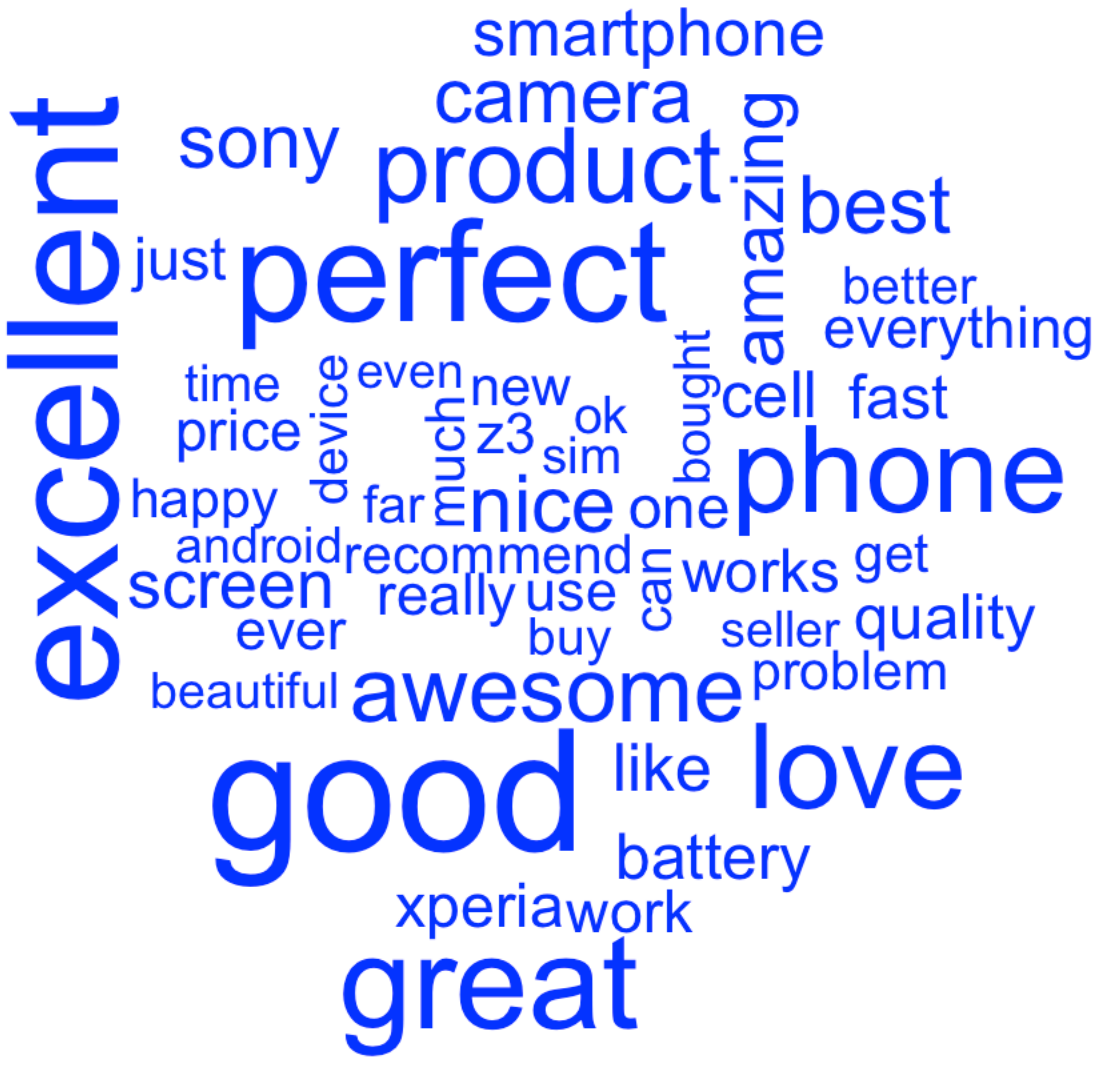
```
## Warning in convert.dfm(x, to = "tm", ...): Argument weighting not used.
```

```
## Warning in weighting(x): empty document(s): 443 594 645 835 913 914 915 916
## 950 1001 1002 1053 1154 1181 1182 1227 1228 1229 1230 1249 1250 1251 1252
## 1253 1254 1255 1256 1257 1258 1259 1260 1261 1262 1263 1264 1265 1266 1267
## 1268 1269 1270 1271 1272 1273
```

```
dtm2 = removeSparseTerms(dtm, 0.97)
dtm2 = as.matrix(dtm2)
#Finding the Frequency
freq = colSums(dtm2)
freq = sort(freq, decreasing = TRUE)


words = names(freq)

wordcloud(words[1:50], freq[1:50], random.color = FALSE, colors = "BLUE")
```


smartphone
camera
sony
product
amazing
best
excellent
just
perfect
better
everything
time
device
even
new
bought
cell
fast
price
much
z3
ok
sim
phone
happy
far
nice
one
android
recommend
can
works
get
screen
really
use
quality
ever
buy
seller
problem
beautiful
awesome
good
like
love
battery
xperia
work
great