Frequent Itemsets

Set the packages. Vital information was hidden as well as warning and output.

```
library(twitteR)

setup_twitter_oauth(key, secret, access, access_secret)

Sys.setenv(JAVA_HOME = "/usr/lib/jvm/default-java")
Sys.setenv(HADOOP_CMD = "/home/bcarancibia/workspace/cuny_msda_is622/hadoop-2.7.1/bin/hadoop")
Sys.setenv(HADOOP_STREAMING = "/home/bcarancibia/workspace/cuny_msda_is622/hadoop-2.7.1/share/hadoop/to

Sys.setenv(SPARK_HOME = "/home/bcarancibia/workspace/cuny_msda_is622/spark-1.4.1-bin-hadoop2.6")
.libPaths(c(file.path(Sys.getenv("SPARK_HOME"), "R", "lib"), .libPaths()))
library(SparkR)

sc <- sparkR.init(master = "local")
sqlContext <- sparkRSQL.init(sc)
```

I am going to look at Twitter data, specifically looking at the hashtag #datarevolution. This is an important hashtag in the international development space because of the recent increase in desire for countries, companies, and aid organizations to integrate analytics into their everyday workflows. I am going to collect hashtags and then count the top ten screen names that use the hashtag #datarevolution.

```
tweets <- searchTwitter("#datarevolution",n=9999)
```

```
## Warning in doRppAPICall("search/tweets", n, params = params,
## retryOnRateLimit = retryOnRateLimit, : 9999 tweets were requested but the
## API can only return 1475
```

```
x <- twListToDF(tweets)

sparkdf <- createDataFrame(sqlContext, x)

group <- agg(group_by(sparkdf, sparkdf$screenName), sum_of_screenname=(count(sparkdf$screenName)))
head(group)
```

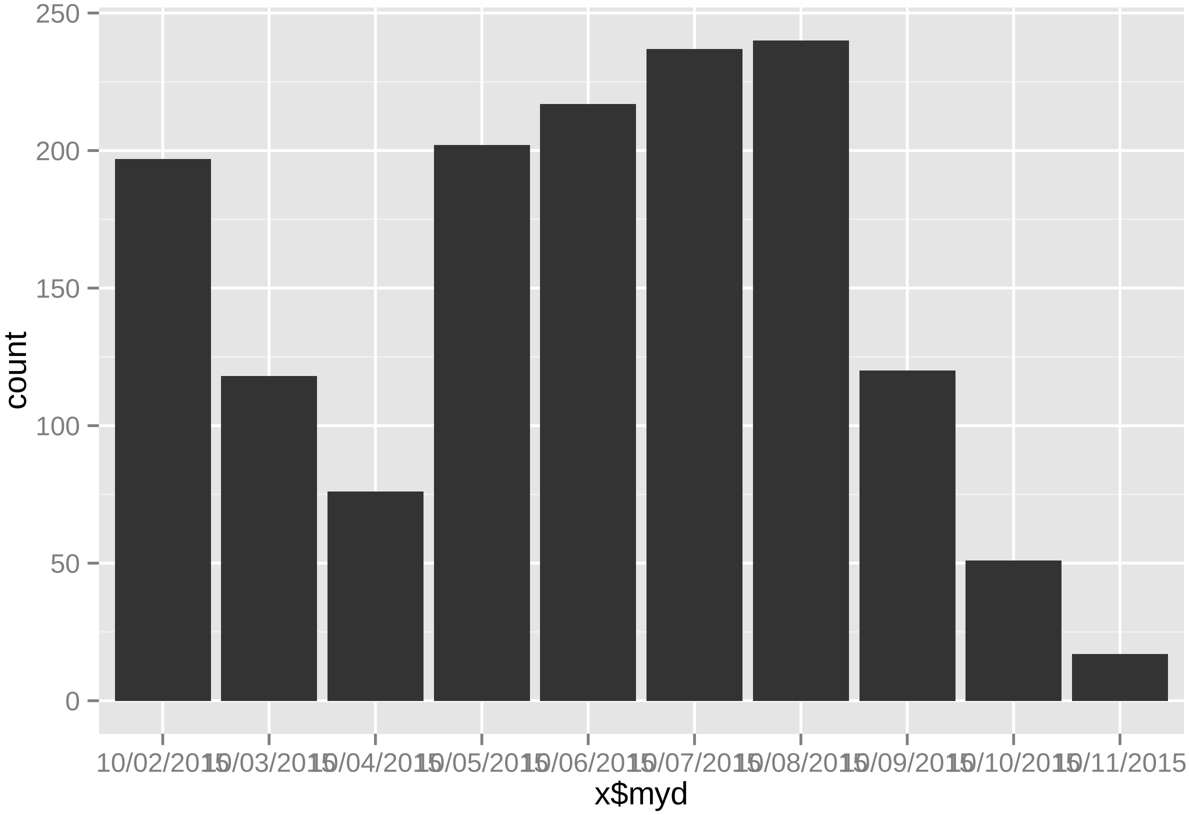
```
##        screenName sum_of_screenname
## 1      blondelena                 6
## 2           fpgil                 1
## 3        keyram10                 3
## 4        alabriqu                 1
## 5 OpenDataService                 2
## 6     bracken10011                1
```

The next step is to parse out dates and then quickly plot the data to get an idea of distribution of the data.

```
library(ggplot2)
library(lubridate)
library(forecast)
```

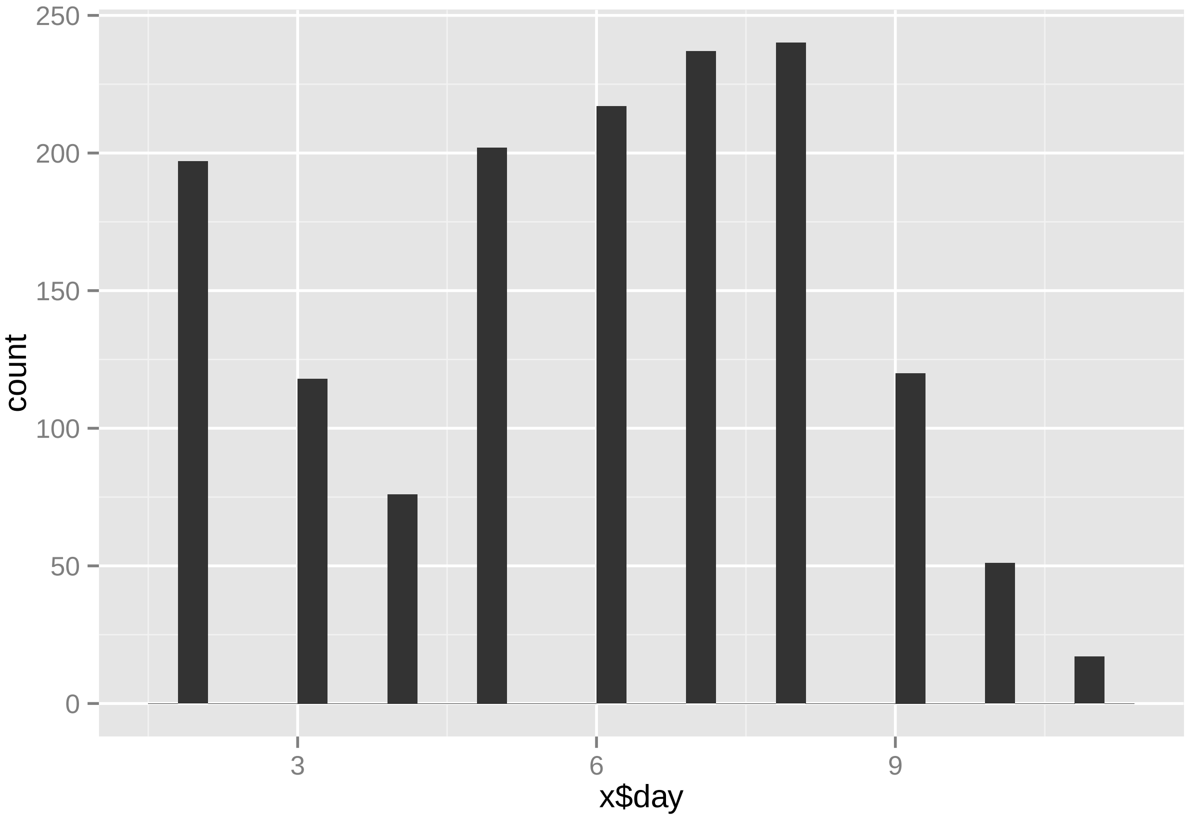
```
## Loading required package: zoo
##
## Attaching package: 'zoo'
##
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
##
## Loading required package: timeDate
## This is forecast 6.1
```

```r
x$created <- parse_date_time(x$created, "%Y%m%d %H%M%S", truncated = 3)

x$day <- day(x$created)
x$month <- month(x$created)
x$year <- year(x$created)
x$hour <- hour(x$created)
x$minute <- minute(x$created)
x$time <- sprintf('%02d:%02d', x$hour, x$minute)
x$myd <- sprintf('%02d/%02d/%02d', x$month, x$day, x$year)

#Tweets by Month, Day, Year
qplot(x$myd, data = x, geom="histogram")
```



```r
qplot(x$day, data=x, geom="histogram")
```

```
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
```



One thing to notice is that the R package to scrape tweets, only seems to take into account the past week or so. Based on the twitteR package, there could be Twitter API restrictions

Group by Screenname and sum the retweets per screen. This can be used in the future to do a social network analysis.

```
group <- agg(group_by(sparkdf, sparkdf$screenName), sum_of_retweets=(sum(sparkdf$retweetCount)))

head(group,10)
```

```
##         screenName sum_of_retweets
## 1       blondelena             119
## 2            fpgil               7
## 3         keyram10              14
## 4         alabriqu               6
## 5  OpenDataService              21
## 6     bracken10011              10
## 7        Cath_Cand               3
## 8        EvarMburu               1
## 9      writeosahon               4
## 10      Scott42195               7
```

Transaction Itemsets.

I decided to look at userid and myd. The hope is to find frequent user ids to specific month year and day.

```
library(arules)
```

```
## Loading required package: Matrix
##
## Attaching package: 'Matrix'
##
## The following objects are masked from 'package:base':
##
##     crossprod, tcrossprod
##
##
## Attaching package: 'arules'
##
## The following objects are masked from 'package:base':
##
##     abbreviate, %in%, write
```

```
transaction <- x[,c('id','myd')]

transactions <- as(split(as.vector(transaction$id),as.vector(transaction$myd)),"transactions")


params <- list(supp=0.2, minlen=2, maxlen=4,target="frequent itemsets")
results <- apriori(transactions,parameter=params)
```

```
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport support minlen maxlen
##         0.8    0.1    1 none FALSE            TRUE     0.2      2      4
##             target   ext
##  frequent itemsets FALSE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## apriori - find association rules with the apriori algorithm
## version 4.21 (2004.05.09)        (c) 1996-2004   Christian Borgelt
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[1475 item(s), 10 transaction(s)] done [0.00s].
## sorting and recoding items ... [0 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 done [0.00s].
## writing ... [0 set(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
results
```

```
## set of 0 itemsets
```

Initial results are 0 whic is surprising, so that means that no userid is tweeting more than once per day. The next implementation of the itemsets is look at userids and hour when tweeting. The hope is to see if there is frequency there.

```
transactions <- x[,c('id','hour')]

transactions <- as(split(as.vector(transactions$id),as.vector(transactions$hour)),"transactions")

params <- list(supp=0.2, minlen=2, maxlen=4,target="frequent itemsets")
results <- apriori(transactions,parameter=params)
```

```
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport support minlen maxlen
##         0.8    0.1    1 none FALSE            TRUE     0.2      2      4
##             target   ext
##  frequent itemsets FALSE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## apriori - find association rules with the apriori algorithm
## version 4.21 (2004.05.09)        (c) 1996-2004   Christian Borgelt
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[1475 item(s), 24 transaction(s)] done [0.00s].
## sorting and recoding items ... [0 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 done [0.00s].
## writing ... [0 set(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
results
```

```
## set of 0 itemsets
```

Again there is a set of 0 itemsets. This might be due to the parameters I set. I will reduce support and min length to see if there is any change.

```
transactions <- x[,c('id','hour')]

transactions <- as(split(as.vector(transactions$id),as.vector(transactions$hour)),"transactions")

params <- list(supp=0.03, minlen=2, maxlen=4,target="frequent itemsets")
results <- apriori(transactions,parameter=params)
```

```
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport support minlen maxlen
##         0.8    0.1    1 none FALSE            TRUE    0.03      2      4
##             target   ext
##  frequent itemsets FALSE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
## Warning in apriori(transactions, parameter = params): You chose a very low absolute support count of
```

```
## apriori - find association rules with the apriori algorithm
## version 4.21 (2004.05.09)        (c) 1996-2004   Christian Borgelt
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[1475 item(s), 24 transaction(s)] done [0.00s].
## sorting and recoding items ... [1475 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 done [0.98s].
## writing ... [30919359 set(s)] done [0.83s].
## creating S4 object  ... done [3.89s].
```

```
results
```

```
## set of 30919359 itemsets
```

Lowering the support level causes an increase in the itemsets.