Utilizing the R Statistical Language for Analyzing Server Logs

Ryan Talabis and Robert McPherson

January 26, 2013

Outline

Processing the Data

Read in the Data

- Data can be read from text files of server logs.
- An R function called scan is useful for this.
- Corpus function is used to convert data into a corpus, consisting of documents and words.
- Each line in the log can be considered a document, so that comparisons can be made across time.

```
> #Read server log - text file
> secureData <- scan("secure.txt", character(0), sep = "\n") # separate each line
> #Save as a word corpus
> data <- Corpus(VectorSource(secureData[1:1000],))
> inspect(data[1:5]) #Inspect the first five entries in the data
```

A corpus with 5 text documents
The metadata consists of 2 tag-value pairs and a data frame Available tags are:
create$_d atecreatorAvailablevariablesinthedataframeare : MetaID$
[[1]] Nov 10 07:53:34 uiftp sshd[28959]:
pam$_u nix(sshd : session) : sessionclosedforuserrichard$
[[2]] Nov 10 08:30:26 uiftp sshd[17793]: Accepted publickey for richard from
10.130.192.162 port 34178 ssh2
[[3]] Nov 10 08:30:26 uiftp sshd[17793]:
pam$_u nix(sshd : session) : sessionopenedforuserrichardby(uid = 0)$
[[4]] Nov 10 08:30:26 uiftp sshd[17793]:
pam$_u nix(sshd : session) : sessionclosedforuserrichard$
[[5]] Nov 10 08:30:29 uiftp sshd[17819]: Accepted publickey for richard from
10.130.192.162 port 34180 ssh2

Function to Trim White Space

- Create a function called, "trim", as a convenient way to remove white space in the data transformation process.
- Utilizes the R function, gsub, and regular expressions.

```
> #Function to strip white space from strings
> trim <- function (x) gsub("^\\s+|\\s+$", "", x)
```

Using Regular Expressions to Parse the Time Stamp

- Regular expressions provide a common sytax for parsing characters and text.
- Regular expressions are strings which locate specific patterns in text.
- These regular expressions extract text patterns which identify time, day, and month from the date stamp in the log file.

```
> #Regular expressions for parsing date information
> regexpTime = "[0-9][0-9]:[0-9][0-9]:[0-9][0-9]"
> regexpDay = "\\s[0-9][0-9]\\s"
> regexpMonth = "^[A-Z][a-z][a-z]\\s"
> time = trim(str_extract(secureData[1:1000],regexpTime))
> day = trim(str_extract(secureData[1:1000],regexpDay))
> month = trim(str_extract(secureData[1:1000],regexpMonth))
```

Results from Parsing the Time Stamp

- Following is a sample from the results extracted from the time stamp.
- The cbind function means, column bind, and binds arrays into columns.
- The data.frame function puts the columnar data into a single table.

```
> #Make a data frame consisting of month, day, and time columns
> timeStampCols = data.frame(cbind(month, day, time))
> timeStampCols[1:5,]

  month day     time
1   Nov  10 07:53:34
2   Nov  10 08:30:26
3   Nov  10 08:30:26
4   Nov  10 08:30:26
5   Nov  10 08:30:29
```

Transforming the Text Corpus with the tm_map Function

- The tm_map function is from the tm text mining package.
- It maps a number of text mining transformations to the data.
- Transformations include stripping whitespace, changing to lower case, removing common stopwords, removing word suffixes - called stemming, removing punctuation, and removing numbers, among others.

Examples of the tm_map Function

```
> data2 = tm_map(data, stripWhitespace)
> data2 = tm_map(data2, tolower)
> stopWords = c(stopwords("english"),"uiftp","sshd")
> data2 = tm_map(data2, removeWords, stopWords)
> data2 = tm_map(data2, stemDocument)
> data2 = tm_map(data2, removePunctuation)
> data2 = tm_map(data2, removeNumbers)
> inspect(data2[1:5])
```

```
A corpus with 5 text documents

The metadata consists of 2 tag-value pairs and a data frame
Available tags are:
  create_date creator
Available variables in the data frame are:
  MetaID

[[1]]
nov     pamunixsession session close  user richard

[[2]]
nov     accept publickey  richard   port  ssh

[[3]]
nov     pamunixsession session   user richard  uid

[[4]]
nov     pamunixsession session close  user richard

[[5]]
nov     accept publickey  richard   port  ssh
```

Making a Document Term Matrix with the DocumentTermMatrix Function

- A document term matrix shows the frequency count of each word.
- Frequencies are shown for each document.
- Documents are in rows.
- Terms, or words, are in columns.

```
> #Make a word frequency matrix, with documents as rows, and terms as columns
> dtm = DocumentTermMatrix(data2)
> inspect(dtm[1:5,1:5])

A document-term matrix (5 documents, 5 terms)

Non-/sparse entries: 2/23
Sparsity           : 92%
Maximal term length: 7
Weighting          : term frequency (tf)

    Terms
Docs accept access address admin age
   1      0      0       0     0   0
   2      1      0       0     0   0
   3      0      0       0     0   0
   4      0      0       0     0   0
   5      1      0       0     0   0
```

Removing Sparse Terms

- It can be useful to remove infrequently occuring words.
- Sometimes the most significant terms are the most fequently occuring ones.
- Use the removeSparseTerms function.

```
> #Remove and inspect sparse terms, with at least 80% sparse occurence
> dtm = removeSparseTerms(dtm, 0.85)
> inspect(dtm[1:5,1:5])

A document-term matrix (5 documents, 5 terms)

Non-/sparse entries: 10/15
Sparsity           : 60%
Maximal term length: 14
Weighting          : term frequency (tf)

    Terms
Docs close invalid nov pamunixsession password
   1     1       0   1              1        0
   2     0       0   1              0        0
   3     0       0   1              1        0
   4     1       0   1              1        0
   5     0       0   1              0        0
```

Transposing the Term Matrix

- Can opt to have the terms in rows, and the documents in columns.
- Use the TermDocumentMatrix for this.
- Same as the DocumentTermMatrix, but transposed.

```
> #Make a word frequency matrix, with terms as rows, and documents as columns
> dtm2 = TermDocumentMatrix(data2)
> inspect(dtm2[1:5,1:5])

A term-document matrix (5 terms, 5 documents)

Non-/sparse entries: 2/23
Sparsity           : 92%
Maximal term length: 7
Weighting          : term frequency (tf)

         Docs
Terms     1 2 3 4 5
  accept  0 1 0 0 1
  access  0 0 0 0 0
  address 0 0 0 0 0
  admin   0 0 0 0 0
  age     0 0 0 0 0
```

Date and Time Stamp Handling

Append Time Element Columns to Text Frequency Columns

- Convert the word corpus to text, so that it can be appended.
- The example below includes a version with all time stamp elements, and another with only the month.

```
> #Add month, day, and time columns to the document term matrix
> #First, turn the dtm into a plain text object
> dtmText = inspect(dtm)
> #This data frame version has the full month, day, and time stamp
> dtmStamp = data.frame(cbind(timeStampCols, dtmText))
> #This data frame version only has the month appended, not day or time
> dtmMonth = data.frame(cbind(month, dtmText))
```

Utilize SQL as One Means of Grouping Data

- Utilize the sqldf function.
- This function allows for writing SQL queries against an R data frame.

```
> #Sum up the frequencies each of the top terms for each month
> topTermsByMonth = sqldf("select month
  ,count(*) as rowCount
  ,sum(user) as user
  ,sum(invalid) as invalid
  ,sum(richard) as richard
  ,sum(port) as port
  ,sum(password) as password
  ,sum(ssh) as ssh
  ,sum(uid) as uid
  ,sum(session) as session
  from dtmMonth group by month")
> topTermsByMonth

  month rowCount user invalid richard port password ssh uid session
1   Apr       36   17       4      21    7        2   5   6      17
2   Dec       16    8       0      10    4        0   2   4       8
3   Feb       54    9       0      37    8       33   2   4       8
4   Jun       91   62      26      20   18       14  18  19      15
5   Mar       22    9       0      18    4        5   4   6       9
6   May      611  328     147     103  121       96 117 119      83
7   Nov      170   97       0      46   43       21  43  44      96
```

Text Analysis

Finding High Frequency Terms

- Frequently occuring words can be identified with the findFreqTerms function
- Can select words that occur within a low and high frequency range
- findFreqTerms(x, lowfreq = 0, highfreq = Inf)
- Can also specify only a single, lower bound, with upper bound defaulting to infinity

```
> #Find the top 20 most frequent terms
> findFreqTerms(dtm, 20)
```

```
 [1] "close"         "invalid"       "nov"           "pamunixsession"
 [5] "password"      "port"          "richard"       "session"
 [9] "ssh"           "uid"           "user"
```

Find Associated Terms

- Can find terms in the text that are associated with a given term.
- Use the findAssocs function
- Can specify the minimum correlation.

```
> #Find associated terms with the term, "root", and a correlation of at least 0.4
> findAssocs(dtm, "session", 0.4)

       session pamunixsession           user          close
          1.00           0.88           0.49           0.40

> findAssocs(dtm, "user", 0.4)

          user        session pamunixsession        invalid
          1.00           0.49           0.45           0.41

> findAssocs(dtm, "port", 0.4)

    port      ssh password
    1.00     0.96     0.62
```

Create a Dictionary

- Create a dictionary specifying a subset of terms to work with
- Use the Dictionary function.

```
> ##Other functions related to selecting specific terms
>
> #Create a dictionary: a subset of terms
> d = Dictionary(c("root", "richard", "invalid", "session"))
> #Use dictionary to make a matrix with only the dictionary terms
> dtm_dictionary = TermDocumentMatrix(data2, list(dictionary = d))
> show(dtm_dictionary)
```

```
A term-document matrix (4 terms, 1000 documents)

Non-/sparse entries: 757/3243
Sparsity           : 81%
Maximal term length: 7
Weighting          : term frequency (tf)
```

Outlier Detection

How to Read Outlier Exhibits

- Each slide has a table containing outlier(s), and a scatterplot graph
- This example code produces one slide for every outlier
- If the occurence frequency of a term happened to be perfectly correlated to the number of log entries for each month, points would line up diagonally on the regression line.
- Outliers are clearly visible, shown as points that are far removed from the regression line.
- Terms shown above the line are those that had more occurences than expected, given the number of log entries for that month; terms below had less.

Outlier Detection Method

- Bonferoni outlier detection method was used to identify outliers.
- The Bonferroni method is less sensitive to false positives than outlier detection based on percentage increases, standard deviations (i.e., z-score), or other methods.
  - Identifying ouliers based on percentage increase thresholds produces far more false positives for small denominator values, and too few for large denominators.
  - The Standard deviation method reduces false positives somewhat, but not enough.
  - The Bonferroni method generally tends to be more balanced across all value ranges.
- The Bonferroni outlier table at the top of each slide contains statistically significant outlier(s).
- The Bonferroni method detects outliers based on t-test of regression coefficients, at the 99.9% confidence level.
- An additional filter was added so that only those states that changed by at least plus or minus 10% can be considered outliers.
- Without outlier detection, thousands of slides would be required to show all the combinations of peril type, business line, and financial division, for each and every data attribute.

Term: invalid

Month	Term	Term Frequency	Total Month's Entries
Nov	invalid	0	170

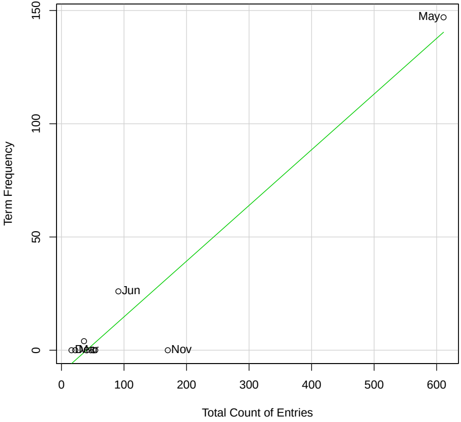

Figure: Comparison of term frequency, invalid, with number of log entries

Term: port

Month	Term	Term Frequency	Total Month's Entries
Nov	port	43	170

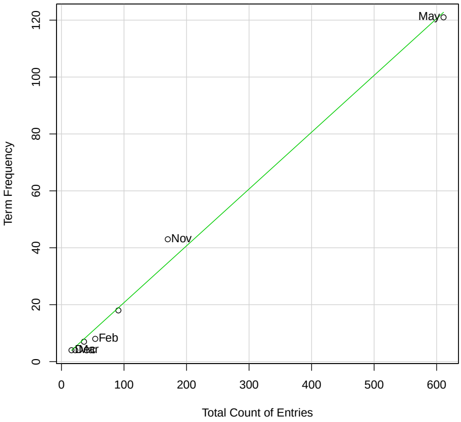

Figure: Comparison of term frequency, port, with number of log entries

Term: password

Month	Term	Term Frequency	Total Month's Entries
Feb	password	33	54

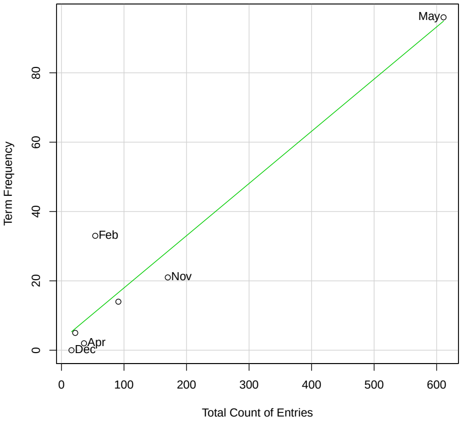

Figure: Comparison of term frequency, password, with number of log entries

Term: session

Month	Term	Term Frequency	Total Month's Entries
May	session	83	611
Nov	session	96	170

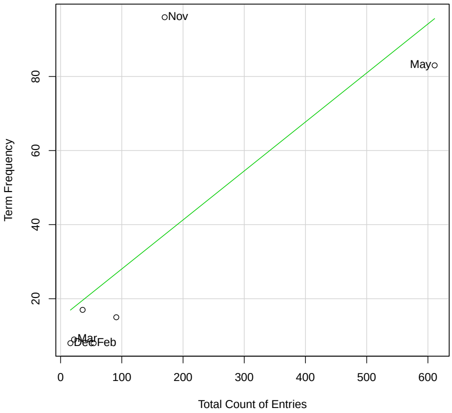

Figure: Comparison of term frequency, session, with number of log entries

Most Significant Outlier Terms and Months

- Can write a loop to summarize the most significant outlier terms in a table
- Same terms as shown in preceding scatterplot graphs

	Month	Term	Term Frequency	Total Month's Entries
2	Nov	invalid	0	170
3	Nov	port	43	170
4	Feb	password	33	54
5	May	session	83	611
6	Nov	session	96	170