Report

Homework for 2015\_datamining\_project1,datamining class of Tsinghua University

计22 滕爽2012011270

目录

[*（一）Document Visualization* 2](#_Toc416473009)

[Q1： 2](#_Toc416473010)

[一、【题目要求】 2](#_Toc416473011)

[二、【设计思路】 2](#_Toc416473012)

[三、【代码实现】 2](#_Toc416473013)

[四、【结果展示】 3](#_Toc416473014)

[Q2： 4](#_Toc416473015)

[一、【题目要求】 4](#_Toc416473016)

[二、【设计思路】 4](#_Toc416473017)

[三、【代码实现】 4](#_Toc416473018)

[四、【结果展示】 5](#_Toc416473019)

[Q3: 6](#_Toc416473020)

[一、【题目要求】 6](#_Toc416473021)

[二、【设计思路】 6](#_Toc416473022)

[三、【代码实现】 6](#_Toc416473023)

[四、【结果展示】 6](#_Toc416473024)

[Q4: 7](#_Toc416473025)

[一、【题目要求】 7](#_Toc416473026)

[二、【设计思路】 7](#_Toc416473027)

[三、【代码实现】 7](#_Toc416473028)

[四、【结果展示】 8](#_Toc416473029)

[五、【结果分析】 11](#_Toc416473030)

[Q5: 11](#_Toc416473031)

[一、【题目要求】 11](#_Toc416473032)

[二、【设计思路】 11](#_Toc416473033)

[三、【代码实现】 11](#_Toc416473034)

[四、【结果展示】 12](#_Toc416473035)

[五、【结果分析】 13](#_Toc416473036)

[Q6: 13](#_Toc416473037)

[一、【题目要求】 13](#_Toc416473038)

[三、【代码实现】 13](#_Toc416473039)

[四、【结果展示】 14](#_Toc416473040)

[五、【结果分析】 15](#_Toc416473041)

[Q7: 15](#_Toc416473042)

[一、【题目要求】 15](#_Toc416473043)

[二、【设计思路】 15](#_Toc416473044)

[三、【代码实现】 15](#_Toc416473045)

[四、【结果展示】 16](#_Toc416473046)

[五、【结果分析】 16](#_Toc416473047)

[（二）Document Similarity 17](#_Toc416473048)

[Q1: 17](#_Toc416473049)

[一、【题目要求】 17](#_Toc416473050)

[二、【设计思路】 17](#_Toc416473051)

[三、【代码实现】 18](#_Toc416473052)

[四、【结果展示】 18](#_Toc416473053)

[Q2: 18](#_Toc416473054)

[一、【题目要求】 18](#_Toc416473055)

[二、【设计思路】 19](#_Toc416473056)

[三、【代码实现】 19](#_Toc416473057)

[四、【结果展示】 21](#_Toc416473058)

[五、【结果分析】 21](#_Toc416473059)

[Q3: 21](#_Toc416473060)

[一、【题目要求】 21](#_Toc416473061)

[二、【设计思路】 21](#_Toc416473062)

[三、【代码实现】 22](#_Toc416473063)

[四、【结果展示】 23](#_Toc416473064)

[五、【结果分析】 24](#_Toc416473065)

# （一）Document Visualization

## Q1：

### 一、【题目要求】

Create document vectors for each of the stories in the folder. Give the commands you used.

### 二、【设计思路】

按照题目的提示阅读“01.R”，查找到read.doc的功能：“Read in an XML news story and extract its full text”。打开并提取xml文件的内容，并返回一个关于文章内容的单词向量。即调用这个函数之后直接返回document vector。

### 三、【代码实现】

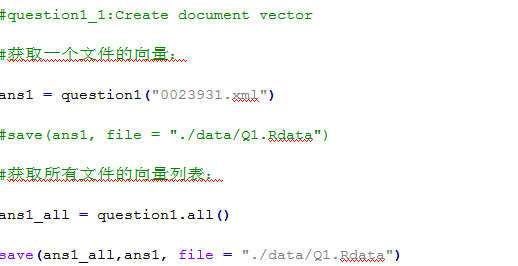
见main.R文件的question1()函数：

#input:filename

#calls:read.doc in 01.R

#output:document vector

运行command.R中：



**Figure 1**

### 四、【结果展示】

数据保存在data/Q1.Rdata中

运行ans1如下图所示：（结果未完全显示）



**Figure 2**

运行ans1\_all如下图所示：



**Figure 3**

## Q2：

### 一、【题目要求】

1）Give a command to extract the 37th word of story number 1595645.xml.

2）Give a command to count the number of times the word “experiencing” appears in that story.

3) Give a command to count the inverse-document-frequency of the word “experiencing”.

### 二、【设计思路】

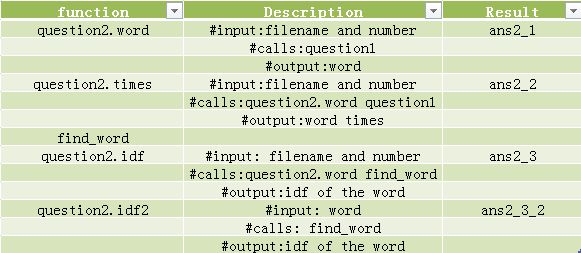
1、Q1提取出单词向量之后直接取下标为37的元素

2、使用R语言自带的table（）函数，取下标为“experiencing”的元素，即为次数。

3、遍历所有文件的单词向量，记录包含有“experiencing”的个数count，代入公式log（文件总数/count）进行计算。

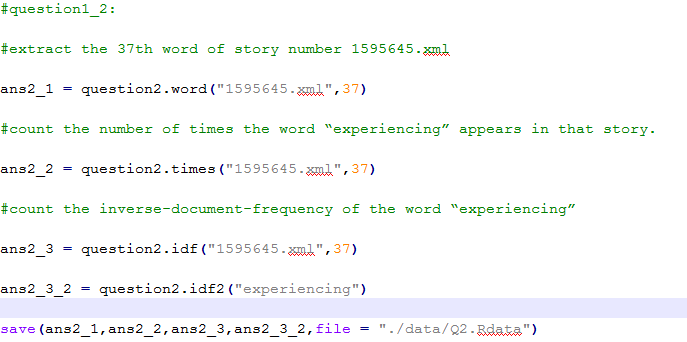
### 三、【代码实现】

见main.R文件下的以下函数：



**Figure 4**

Command.R中：



**Figure 5**

### 四、【结果展示】

数据保存在data/Q2.Rdata中

如下图所示：



**Figure 6**



**Figure 7**



**Figure 8**

## Q3:

### 一、【题目要求】

Give the commands you would use to construct a bag-of-words data-frame from the document vectors for the stories.

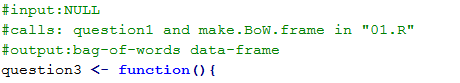
### 二、【设计思路】

提取所有文件的单词向量，每个单词向量都用table生成列联函数，将所有的列联函数构成列表。

使用“01.R”中的make .BoW.frame函数，自动生成data-frame。

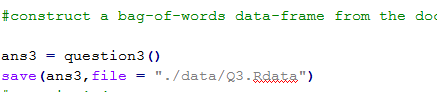
### 三、【代码实现】

见main.R文件下函数：



**Figure 9**

Command.R中：

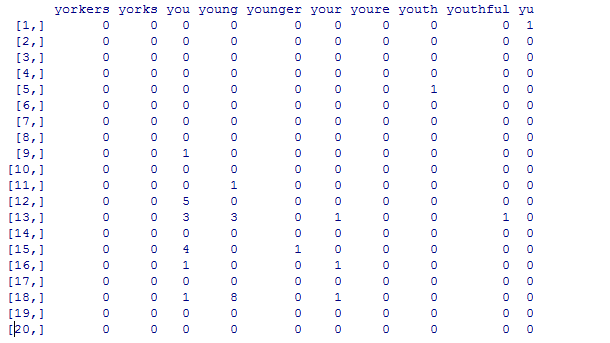


**Figure 10**

### 四、【结果展示】

数据保存在data/Q3.Rdata中

如下图所示：（结果未完全显示）



**Figure 11**

## Q4:

### 一、【题目要求】

Visualize the distribution of word length and give analysis. Give the commands you used. Hint: Using hist.

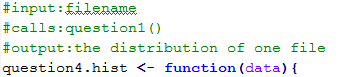
### 二、【设计思路】

总的单词长度分布：统计每个文章的单词向量，将其连接成一个总的单词向量，使用table函数统计数量，代入到hist函数中作图。

实现了两个函数一个为单个文章单词长度分布，一个为总的词长分布。

### 三、【代码实现】

见main.R文件下函数：

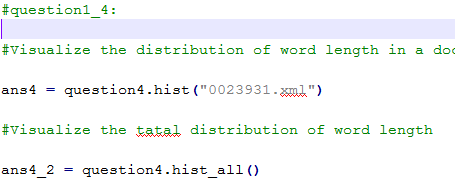


**Figure 12**



**Figure 13**

Command.R中：

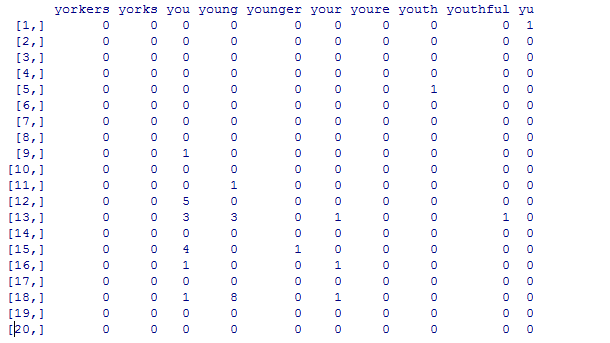
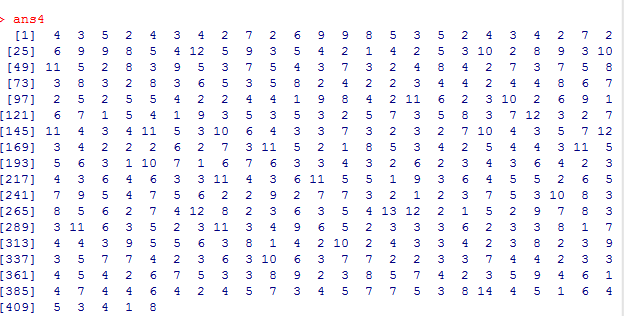


**Figure 14**

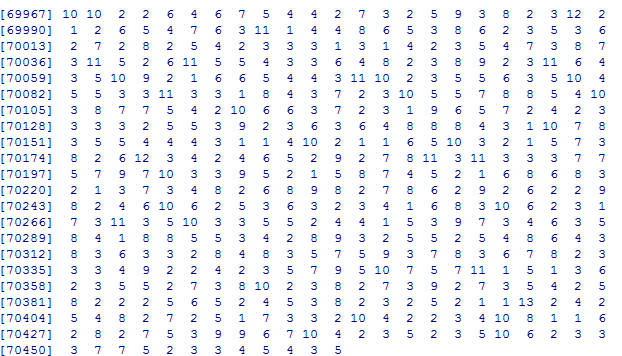
### 四、【结果展示】

数据保存在data/Q4.Rdata中

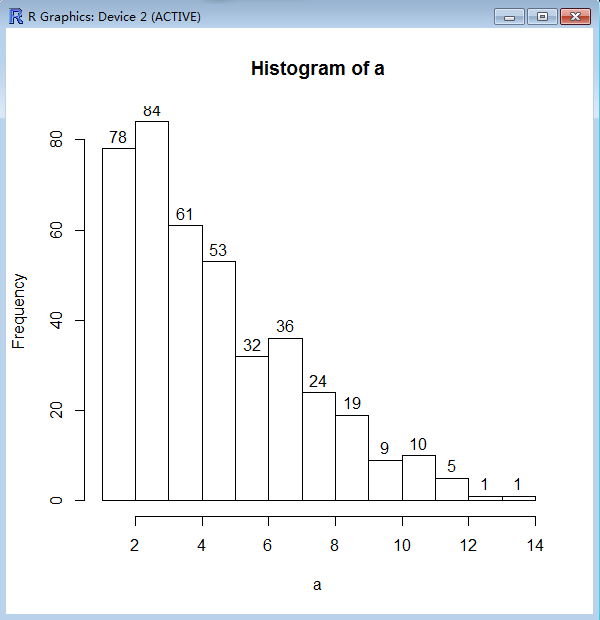
如下图所示：（结果未完全显示）



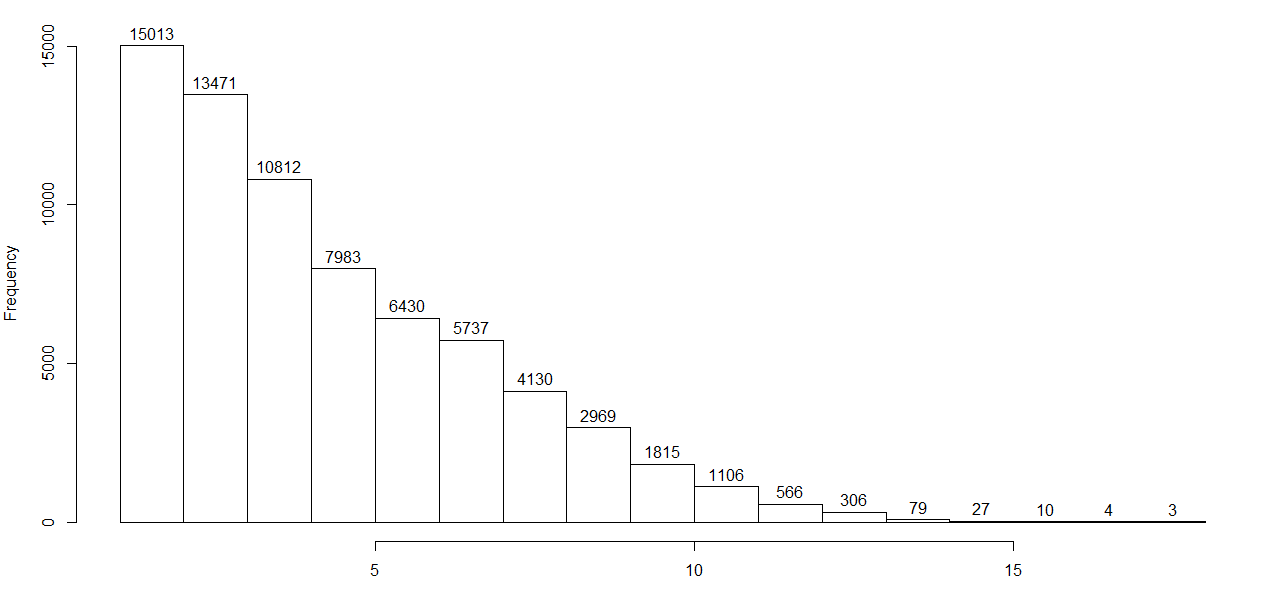
**Figure 15**



**Figure 16**



**Figure 17**



**Figure 18**

### 五、【结果分析】

从总体单词长度分布上来看，整体呈现递减趋势，长度越短，频率越高。说明使用的词汇短的居多，长的比较少。原因可能是比较方便拼写和记忆，便捷性强。

## Q5:

### 一、【题目要求】

Visualize the number of documents in each category and give analysis. Give the commands you used

### 二、【设计思路】

使用xml包提取



**Figure 19**

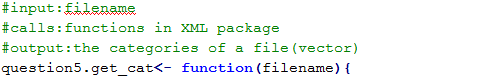
所规定的内容，因为一个文件可能属于不同类别，且类别A可能为类别B的根目录，所以在此处对类别互相包含的进行合并，留下类别路径较长的类别，即为相对子目录的类，从而统计出某文件属于某几个互异的类别。

将所有文件属于的类向量连接成总的类向量，同样使用列联函数table进行数量的统计。

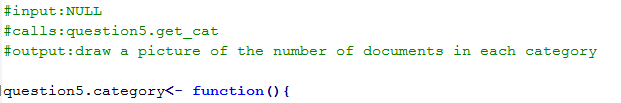
最后代入到barplot函数中进行作图。

### 三、【代码实现】

见main.R文件下函数：

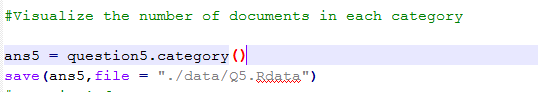


**Figure 20**



**Figure 21**

Command.R中的指令：



**Figure 22**

### 四、【结果展示】

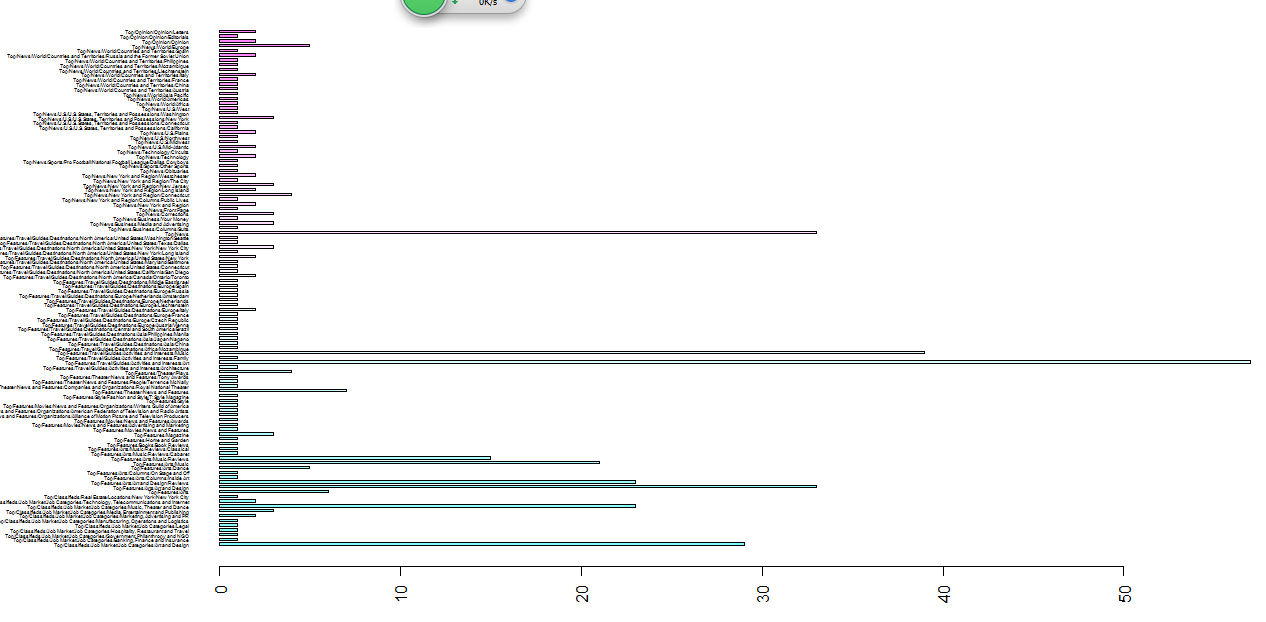
数据保存在data/Q5.Rdata中

如下图所示：（结果未完全显示）



**Figure 23**

见category.pdf



**Figure 24**

### 五、【结果分析】

参见category.pdf。

从分布图上看，由于一篇文章可能属于不同的类型，文章所属类型较多，在数据的展现上面难度比较大。每种类型文章的数目不是很均匀，参差不齐。含有较多数量类别较少。

大部分的类型包含文章数量较少，几乎不超过5，说明文章类型之间确实存在一定的区分度。

最多的类型是Top/Features/Travel/Guides/Activities and Interests/Art，数量超过70，而文章的总数有102，说明大部分的文章都涵盖艺术的类型。

第二多的类型为Top/Features/Travel/Guides/Activities and Interests/Music，从结果上来看，音乐也属于艺术的范畴。因此可以推断出，以艺术类为标准分类文章可能并非有效的分割方法，若想得到好的效果，可能需要其他的类型标准进行分类。

## Q6:

### 一、【题目要求】

Visualize the number of documents in each month and give analysis. Give the commands you used二、【设计思路】

使用xml包提取

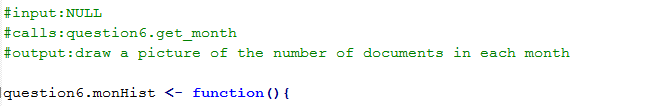


**Figure 25**

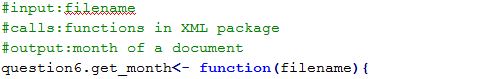
的content，即为月份，将所有文件的月份连成向量之后，使用列联函数table统计向量中各个元素个数，最后barplot进行画图。

### 三、【代码实现】

见main.R文件下函数：

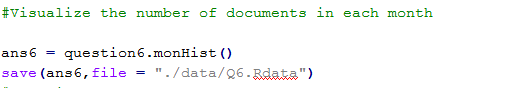


**Figure 26**



**Figure 27**

Command.R中的指令：

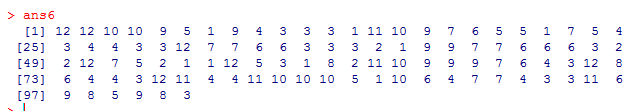


**Figure 28**

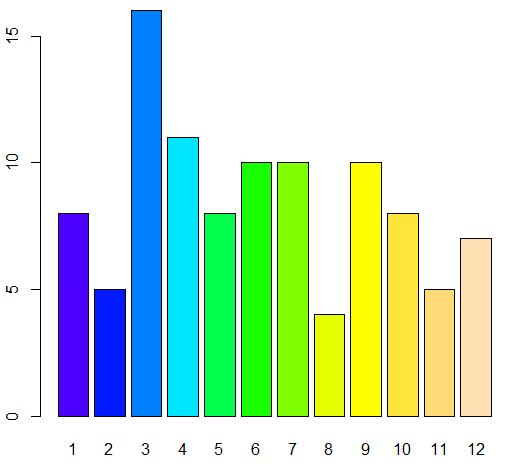
### 四、【结果展示】

数据保存在data/Q6.Rdata中

如下图所示：



**Figure 29**



**Figure 30**

### 五、【结果分析】

出版最多的月份为3月份，最少的月份为8月份，但是2,8月份相差不大，最高与最低相差3倍左右。总体上来讲，除3月份以外的其他月份，数量都比较均匀。猜测3月份可能为发表的截止日期，所以数量较大。

## Q7:

### 一、【题目要求】

Visualize the number of documents in each location and give analysis.

### 二、【设计思路】

使用xml包提取



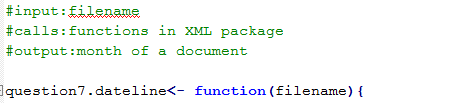
**Figure 31**

的内容，注意此处需要将地点分割出来。有的文件没有这个field。

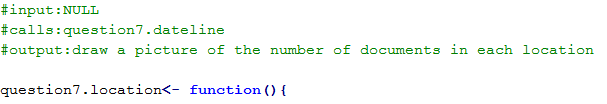
将所有文件的地点连成向量之后，使用列联函数table统计向量中各个元素个数，最后barplot进行画图。

### 三、【代码实现】

见main.R文件下函数：

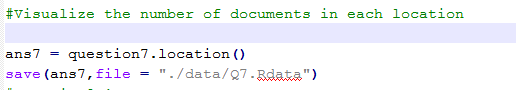


**Figure 32**



**Figure 33**

Command.R中的指令：

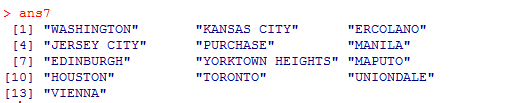


**Figure 34**

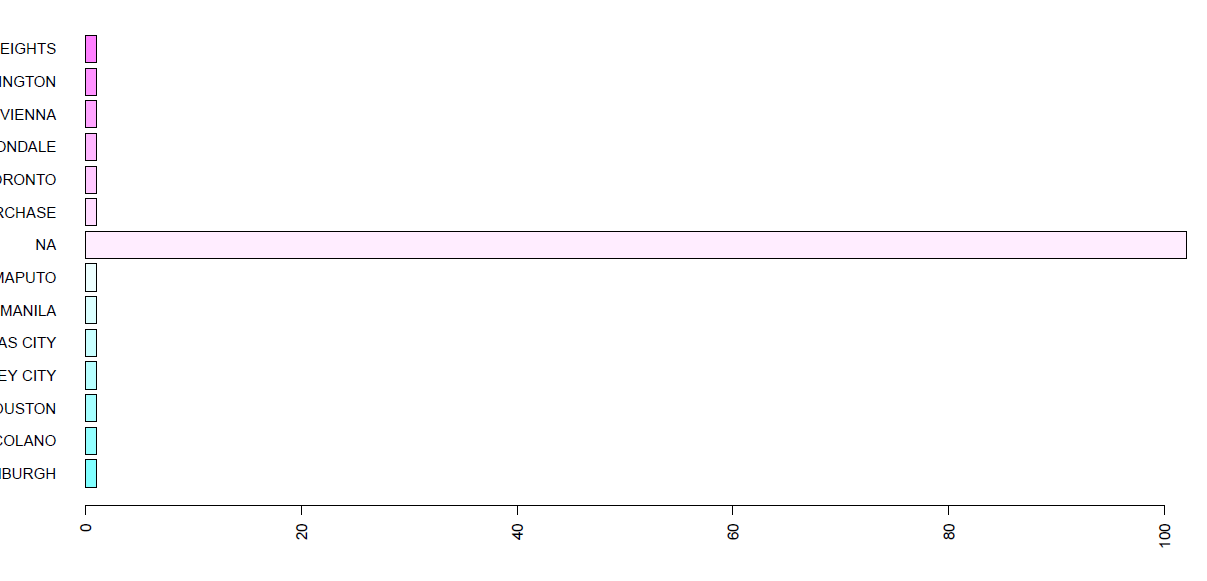
### 四、【结果展示】

数据保存在data/Q7.Rdata中

如下图所示：



**Figure 35**



**Figure 36**

### 五、【结果分析】

结果上来看，有location的文章非常少，但是有的每个地方的数量相同且很少，为1，NA的意思是空，文章数量达到115个。有地址的文章有13个。

# （二）Document Similarity

## Q1:

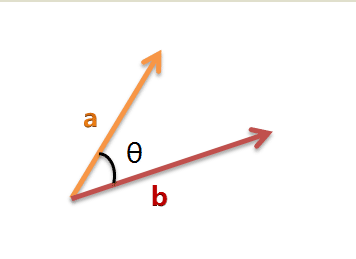
### 一、【题目要求】

Create distance matrices from this data frame for the cosine “distance” (cosine similarity) using the document vectors. Give the commands you use.

### 二、【设计思路】

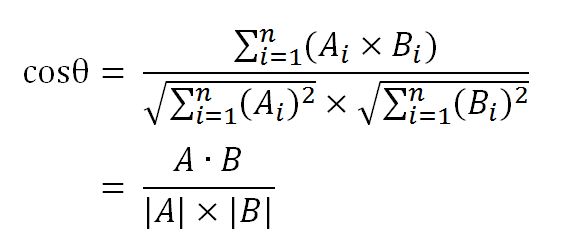
考虑利用第一大题第三小题生成的bag-of-words data-frame。取文件a，b对应data-frame的两行。

把它们想象成空间中的两条线段，都是从原点（[0, 0, ...]）出发，指向不同的方向。两条线段之间形成一个夹角，如果夹角为0度，意味着方向相同、线段重合；如果夹角为90度，意味着形成直角，方向完全不相似；如果夹角为180度，意味着方向正好相反。**因此，我们可以通过夹角的大小，来判断向量的相似程度。夹角越小，就代表越相似。**



**Figure 37**

假定A和B是两个n维向量，A是 [A1, A2, ..., An] ，B是 [B1, B2, ..., Bn] ，则A与B的夹角θ的余弦等于：

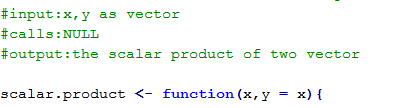
****

**Figure 38**

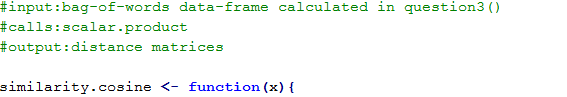
通过上述步骤即可得到a，b之间的夹角的cosine值，填充进矩阵的第a行第b列，从而得到该相似矩阵。

### 三、【代码实现】

见main.R文件下函数：

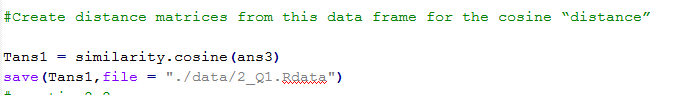


**Figure 39**



**Figure 40**

Command.R中的指令：

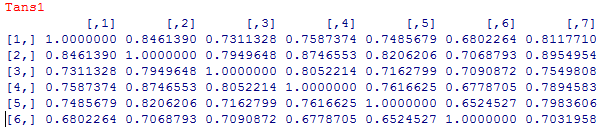


**Figure 41**

### 四、【结果展示】

数据保存在data/2\_Q1.Rdata中（结果未完全显示）

如下图所示：



**Figure 42**

## Q2:

### 一、【题目要求】

Calculate the average distance between stories in the same category and between stories in different categories

### 二、【设计思路】

通过第一题的第5问得到了提取某个文件属于的所有的互异类的函数。

首先遍历所有文件，统计出每个文件属于的类，将这个结果抽象成一个矩阵，矩阵的行为文件名，矩阵的列为类，若该文件i属于该类j，则矩阵A的第i行，第j列，填充1，否则填充0。

其次，取出矩阵A的不同两行a,b对比，若完全相同，说明文件a,b所属于的类相同，将这两个文件划归为1类，采用该方法，找到所有与a，b相同的类，全部归为1类。

采用上面的方法，遍历所有可能的两两文件组合，将所有的文件分成n类(n >= 1)。

最后，构建文件与n类的对应矩阵B。

该矩阵的行为文件号，列为某一个属于n的类。若某文件i，属于n类中的某类j，将B的第i行，第j列填充1，否则填0。

下面利用矩阵B计算同类平均值。

考虑利用第2题第1问的余弦相似矩阵简称sc。

取出某列col为1的所有行号，构成向量x。遍历x的所有两两组合ij，将sc[i.j]的值累加起来得到sum[col]。遍历所有的列，将所有sum[col]的值累加起来，得到sum。利用排列组合公式，长度为n的序列，有n（n-1）/2种不同的组合方式，得到所有两两组合ij的对数count。

最终的结果为sum/count。

下面利用矩阵B计算异类平均值。

取出某列col\_1为1的所有行号，构成向量x。

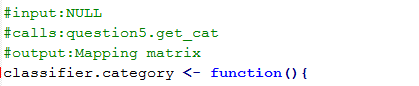
取出某列col\_2为1的所有行号，构成向量y。

遍历x,y中各取一点的所有两两组合ij，将sc[i.j]的值累加起来得到sum。利用排列组合公式，有length（x）乘以length（y）种不同的组合方式，得到所有两两组合ij的对数count。

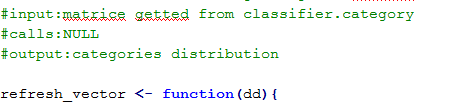
最终的结果为sum/count。

### 三、【代码实现】

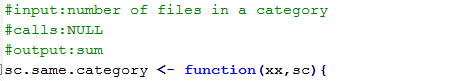
见main.R文件下函数：



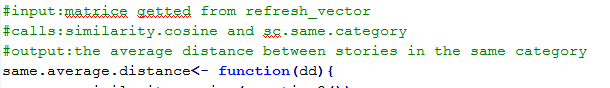
**Figure 43**



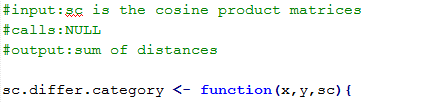
**Figure 44**



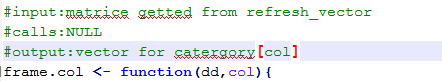
**Figure 45**



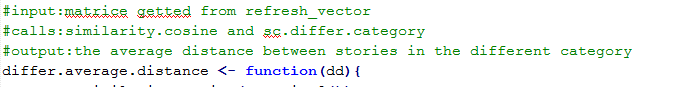
**Figure 46**



**Figure 47**

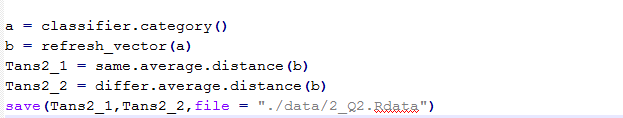


**Figure 48**



**Figure 49**

Command.R中的指令：



**Figure 50**

### 四、【结果展示】

数据保存在data/2\_Q2.Rdata中

如下图所示：



**Figure 51**

### 五、【结果分析】

从结果上面看，不同类别之间的相似度值小于相同类别之间的相似度，说明不同类别之间的词的差异性比较大，同时也证明了余弦相似性是判断文章类型的有效手段。

## Q3:

### 一、【题目要求】

Write a function to find the document which best matches a given query string. The function should take two arguments:

— The query, as a single character string

— The bag-of-words matrix

### 二、【设计思路】

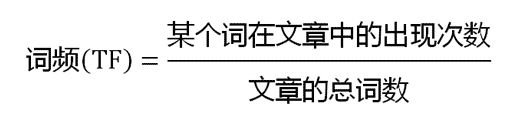
首先提取query的单词向量x。

将向量提取出来有几点注意，需要删除一部分多余的词。X中的某些单词可能从来没有在Data-frame中出现过，考虑其无效性，将其删除。

将x的维数整合成与data-frame中列的维数（即词数）相同。X缺少的单词频度计0。

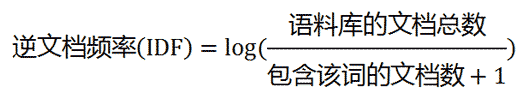
计算TF-IDF权重矩阵原理。

**第一步，计算词频。**



**Figure 52**

**第二步，计算逆文档频率。**

****

**Figure 53**

**第三步，计算TF-IDF。**

****

**Figure 54**

下面举例说明TF-IDF权重矩阵的第i行第j列如何计算。

aa为data-frame，TF[i,j]为词频矩阵。

TF[i,j] <- aa[i,j] / aa.rowsum[i]

IDF <- log(nrow(aa)/(aa.colsum+1))

TFIDF[,j] <- TF[,j] \* IDF[j]

计算权重矩阵之后，分别将query和data-frame进行加权。这里将query的向量query.new视作一个文档。

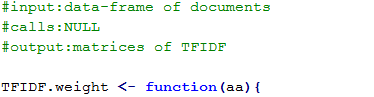




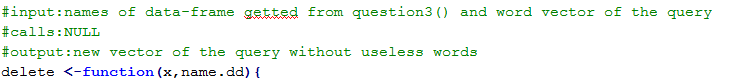
将加权后的向量，代入nearest.points函数，即可返回相似度最高的文章序号和最相似的距离。

### 三、【代码实现】

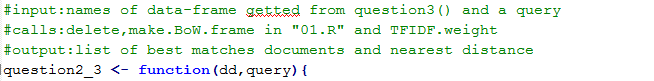
见main.R文件下函数：



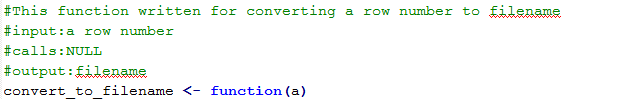
**Figure 55**



**Figure 56**

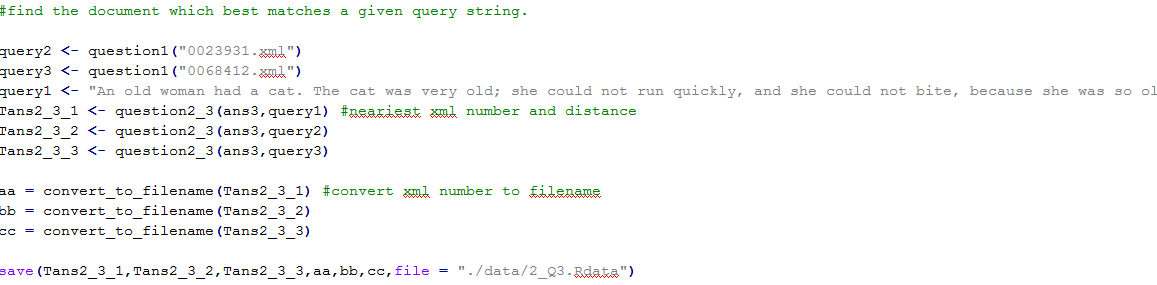


**Figure 57**



**Figure 58**

Command.R中的指令：

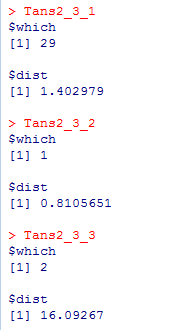


**Figure 59**

### 四、【结果展示】

数据保存在data/2\_Q3.Rdata中

如下图所示：



**Figure 60**

### 五、【结果分析】

从结果上面来看，验证两个文件所属类型结果正确预测，说明此种预测方法较为合理。

同时也证明了TF-DTF权重的对于发掘文档相似性的有效性，可以取得良好的效果，**某个词对文章的重要性越高，它的TF-IDF值就越大。所以，排在最前面的几个词，就是这篇文章的关键词**。TF-IDF算法的优点是简单快速，结果比较符合实际情况。缺点是，单纯以"词频"衡量一个词的重要性，不够全面，有时重要的词可能出现次数并不多。而且，这种算法无法体现词的位置信息，出现位置靠前的词与出现位置靠后的词，都被视为重要性相同，这是不正确的。