REPORT

Homework for
2015_datamining_project1,datamining class of Tsinghua University

计 22
滕爽
2012011270

目录

（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：

```
#question1_1:Create document vector

#获取一个文件的向量：

ans1 = question1("0023931.xml")

#save(ans1, file = "./data/Q1.Rdata")

#获取所有文件的向量列表：

ans1_all = question1.all()

save(ans1_all,ans1, file = "./data/Q1.Rdata")
```

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）

```
> ans1
  [1] "lead"          "the"           "music"         "of"
  [5] "east"          "and"           "west"          "is"
  [9] "divided"       "by"            "common"        "materials"
 [13] "following"     "opposite"      "paths"         "the"
 [17] "music"         "of"            "east"          "and"
 [21] "west"          "is"            "divided"       "by"
 [25] "common"        "materials"     "following"     "opposite"
 [29] "paths"         "thus"          "contemporary"  "asian"
 [33] "composers"     "are"           "often"         "torn"
 [37] "by"            "a"             "wish"          "to"
 [41] "enter"         "the"           "mainstream"    "of"
 [45] "european"      "tradition"     "its"           "particular"
 [49] "instruments"   "forms"         "of"            "rhetoric"
 [53] "and"           "attitudes"     "about"         "how"
 [57] "musical"       "ideas"         "grow"          "and"
 [61] "develop"       "and"           "an"            "even"
 [65] "stronger"      "need"          "to"            "protect"
 [69] "and"           "promote"       "their"         "cultures"
 [73] "own"           "specific"      "use"           "of"
 [77] "interval"      "and"           "rhythm"        "given"
 [81] "the"           "cross"         "purposes"      "at"
```

Figure 2

运行 ans1_all 如下图所示：

```
[1144] "#"              "autobiography"  "my"
[1147] "life"           "in"             "toons"
[1150] "from"           "flatbush"       "to"
[1153] "bedrock"        "in"             "under"
[1156] "a"              "century"        "the"
[1159] "way"            "to"             "appreciate"
[1162] "them"           "is"             "to"
[1165] "see"            "them"           "correction"
[1168] "december"       "#"              "#"
[1171] "friday"         "an"             "obituary"
[1174] "on"             "tuesday"        "about"
[1177] "the"            "animation"      "innovator"
[1180] "joseph"         "barbera"        "misspelled"
[1183] "part"           "of"             "the"
[1186] "title"          "of"             "the"
[1189] "first"          "television"     "series"
[1192] "by"             "him"            "and"
[1195] "his"            "partner"        "william"
[1198] "hanna"          "it"             "was"
[1201] "the"            "ruff"           "reddy"
[1204] "show"           "not"            "ready"
```

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：

function	Description	Result
question2.word	#input:filename and number	ans2_1
	#calls:question1	
	#output:word	
question2.times	#input:filename and number	ans2_2
	#calls:question2.word question1	
	#output:word times	
find_word		
question2.idf	#input: filename and number	ans2_3
	#calls:question2.word find_word	
	#output:idf of the word	
question2.idf2	#input: word	ans2_3_2
	#calls: find_word	
	#output:idf of the word	

Figure 4

Command.R 中：

```
#question1_2:

#extract the 37th word of story number 1595645.xml

ans2_1 = question2.word("1595645.xml",37)

#count the number of times the word "experiencing" appears in that story.

ans2_2 = question2.times("1595645.xml",37)

#count the inverse-document-frequency of the word "experiencing"

ans2_3 = question2.idf("1595645.xml",37)

ans2_3_2 = question2.idf2("experiencing")

save(ans2_1,ans2_2,ans2_3,ans2_3_2,file = "./data/Q2.Rdata")
```

Figure 5

四、【结果展示】

数据保存在 data/Q2.Rdata 中
如下图所示：

```
> ans2_1
[1] "experiencing"
```

Figure 6

```
> ans2_2
experiencing
           1
> ans2_3
[1] 4.624973
```

Figure 7

```
> ans2_3_2
[1] 4.624973
```

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：

```
#input:NULL
#calls: question1 and make.BoW.frame in "01.R"
#output:bag-of-words data-frame
question3 <- function(){
```

Figure 9

Command.R 中：

```
#construct a bag-of-words data-frame from the doc

ans3 = question3()
save(ans3,file = "./data/Q3.Rdata")
```

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）

	yorkers	yorks	you	young	younger	your	youre	youth	youthful	yu
[1,]	0	0	0	0	0	0	0	0	0	1
[2,]	0	0	0	0	0	0	0	0	0	0
[3,]	0	0	0	0	0	0	0	0	0	0
[4,]	0	0	0	0	0	0	0	0	0	0
[5,]	0	0	0	0	0	0	0	1	0	0
[6,]	0	0	0	0	0	0	0	0	0	0
[7,]	0	0	0	0	0	0	0	0	0	0
[8,]	0	0	0	0	0	0	0	0	0	0
[9,]	0	0	1	0	0	0	0	0	0	0
[10,]	0	0	0	0	0	0	0	0	0	0
[11,]	0	0	0	1	0	0	0	0	0	0
[12,]	0	0	5	0	0	0	0	0	0	0
[13,]	0	0	3	3	0	1	0	0	1	0
[14,]	0	0	0	0	0	0	0	0	0	0
[15,]	0	0	4	0	1	0	0	0	0	0
[16,]	0	0	1	0	0	1	0	0	0	0
[17,]	0	0	0	0	0	0	0	0	0	0
[18,]	0	0	1	8	0	1	0	0	0	0
[19,]	0	0	0	0	0	0	0	0	0	0
[20,]	0	0	0	0	0	0	0	0	0	0

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：

```
#input:filename
#calls:question1()
#output:the distribution of one file
question4.hist <- function(data){
```

Figure 12

```
question4.hist_all <- function(){
```

Figure 13

Command.R 中：

```
#question1_4:

#Visualize the distribution of word length in a doc

ans4 = question4.hist("0023931.xml")

#Visualize the tatal distribution of word length

ans4_2 = question4.hist_all()
```

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）

```
> ans4
  [1]  4  3  5  2  4  3  4  2  7  2  6  9  9  8  5  3  5  2  4  3  4  2  7  2
 [25]  6  9  9  8  5  4 12  5  9  3  5  4  2  1  4  2  5  3 10  2  8  9  3 10
 [49] 11  5  2  8  3  9  5  3  7  5  4  3  7  3  2  4  8  4  2  7  3  7  5  8
 [73]  3  8  3  2  8  3  6  5  3  5  8  2  4  2  2  3  4  4  2  4  4  8  6  7
 [97]  2  5  2  5  5  4  2  2  4  4  1  9  8  4  2 11  6  2  3 10  2  6  9  1
[121]  6  7  1  5  4  1  9  3  5  3  5  3  2  5  7  3  5  8  3  7 12  3  2  7
[145] 11  4  3  4 11  5  3 10  6  4  3  3  7  3  2  3  2  7 10  4  3  5  7 12
[169]  3  4  2  2  2  6  2  7  3 11  5  2  1  8  5  3  4  2  5  4  4  3 11  5
[193]  5  6  3  1 10  7  1  6  7  6  3  3  4  3  2  6  2  3  4  3  6  4  2  3
[217]  4  3  6  4  6  3  3 11  4  3  6 11  5  5  1  9  3  6  4  5  5  2  6  5
[241]  7  9  5  4  7  5  6  2  2  9  2  7  7  3  2  1  2  3  7  5  3 10  8  3
[265]  8  5  6  2  7  4 12  8  2  3  6  3  5  4 13 12  2  1  5  2  9  7  8  3
[289]  3 11  6  3  5  2  3 11  3  4  9  6  5  2  3  3  3  6  2  3  3  8  1  7
[313]  4  4  3  9  5  5  6  3  8  1  4  2 10  2  4  3  3  4  2  3  8  2  3  9
[337]  3  5  7  7  4  2  3  6  3 10  6  3  7  7  2  2  3  3  7  4  4  2  3  3
[361]  4  5  4  2  6  7  5  3  3  8  9  2  3  8  5  7  4  2  3  5  9  4  6  1
[385]  4  7  4  4  6  4  2  4  5  7  3  4  5  7  7  5  3  8 14  4  5  1  6  4
[409]  5  3  4  1  8
```

	yorkers	yorks	you	young	younger	your	youre	youth	youthful	yu
[1,]	0	0	0	0	0	0	0	0	0	1
[2,]	0	0	0	0	0	0	0	0	0	0
[3,]	0	0	0	0	0	0	0	0	0	0
[4,]	0	0	0	0	0	0	0	0	0	0
[5,]	0	0	0	0	0	0	0	1	0	0
[6,]	0	0	0	0	0	0	0	0	0	0
[7,]	0	0	0	0	0	0	0	0	0	0
[8,]	0	0	0	0	0	0	0	0	0	0
[9,]	0	0	1	0	0	0	0	0	0	0
[10,]	0	0	0	0	0	0	0	0	0	0
[11,]	0	0	0	1	0	0	0	0	0	0
[12,]	0	0	5	0	0	0	0	0	0	0
[13,]	0	0	3	3	0	1	0	0	1	0
[14,]	0	0	0	0	0	0	0	0	0	0
[15,]	0	0	4	0	1	0	0	0	0	0
[16,]	0	0	1	0	0	1	0	0	0	0
[17,]	0	0	0	0	0	0	0	0	0	0
[18,]	0	0	1	8	0	1	0	0	0	0
[19,]	0	0	0	0	0	0	0	0	0	0
[20,]	0	0	0	0	0	0	0	0	0	0

Figure 15

[69967]	10	10	2	2	6	4	6	7	5	4	4	2	7	3	2	5	9	3	8	2	3	12	2
[69990]	1	2	6	5	4	7	6	3	11	1	4	4	8	6	5	3	8	6	2	3	5	3	6
[70013]	2	7	2	8	2	5	4	2	3	3	3	1	3	1	4	2	3	5	4	7	3	8	7
[70036]	3	11	5	2	6	11	5	5	4	3	3	6	4	8	2	3	8	9	2	3	11	6	4
[70059]	3	5	10	9	2	1	6	6	5	4	4	3	11	10	2	3	5	5	6	3	5	10	4
[70082]	5	5	3	3	11	3	3	1	8	4	3	7	2	3	10	5	5	7	8	8	5	4	10
[70105]	3	8	7	7	5	4	2	10	6	6	3	7	2	3	1	9	6	5	7	2	4	2	3
[70128]	3	3	3	2	5	5	3	9	2	3	6	3	6	4	8	8	8	4	3	1	10	7	8
[70151]	3	5	5	4	4	4	3	1	1	4	10	2	1	1	6	5	10	3	2	1	5	7	3
[70174]	8	2	6	12	3	4	2	4	6	5	2	9	2	7	8	11	3	11	3	3	3	7	7
[70197]	5	7	9	7	10	3	3	9	5	2	1	5	8	7	4	5	2	1	6	8	6	8	3
[70220]	2	1	3	7	3	4	8	2	6	8	9	8	2	7	8	6	2	9	2	6	2	2	9
[70243]	8	2	4	6	10	6	2	5	3	6	3	2	3	4	1	6	8	3	10	6	2	3	1
[70266]	7	3	11	3	5	10	3	3	5	5	2	4	4	1	5	3	9	7	3	4	6	3	5
[70289]	8	4	1	8	8	5	5	3	4	2	8	9	3	2	5	5	2	5	4	8	6	4	3
[70312]	8	3	6	3	3	2	8	4	8	3	5	7	5	9	3	7	8	3	6	7	8	2	3
[70335]	3	3	4	9	2	2	4	2	3	5	7	9	5	10	7	5	7	11	1	5	1	3	6
[70358]	2	3	5	5	2	7	3	8	10	2	3	8	2	7	3	9	2	7	3	5	4	2	5
[70381]	8	2	2	2	5	6	5	2	4	5	3	8	2	3	2	5	2	1	1	13	2	4	2
[70404]	5	4	8	2	7	2	5	1	7	3	3	2	10	4	2	2	3	4	10	8	1	1	6
[70427]	2	8	2	7	5	3	9	9	6	7	10	4	2	3	5	2	3	5	10	6	2	3	3
[70450]	3	7	7	5	2	3	3	4	5	4	3	5											

Figure 16

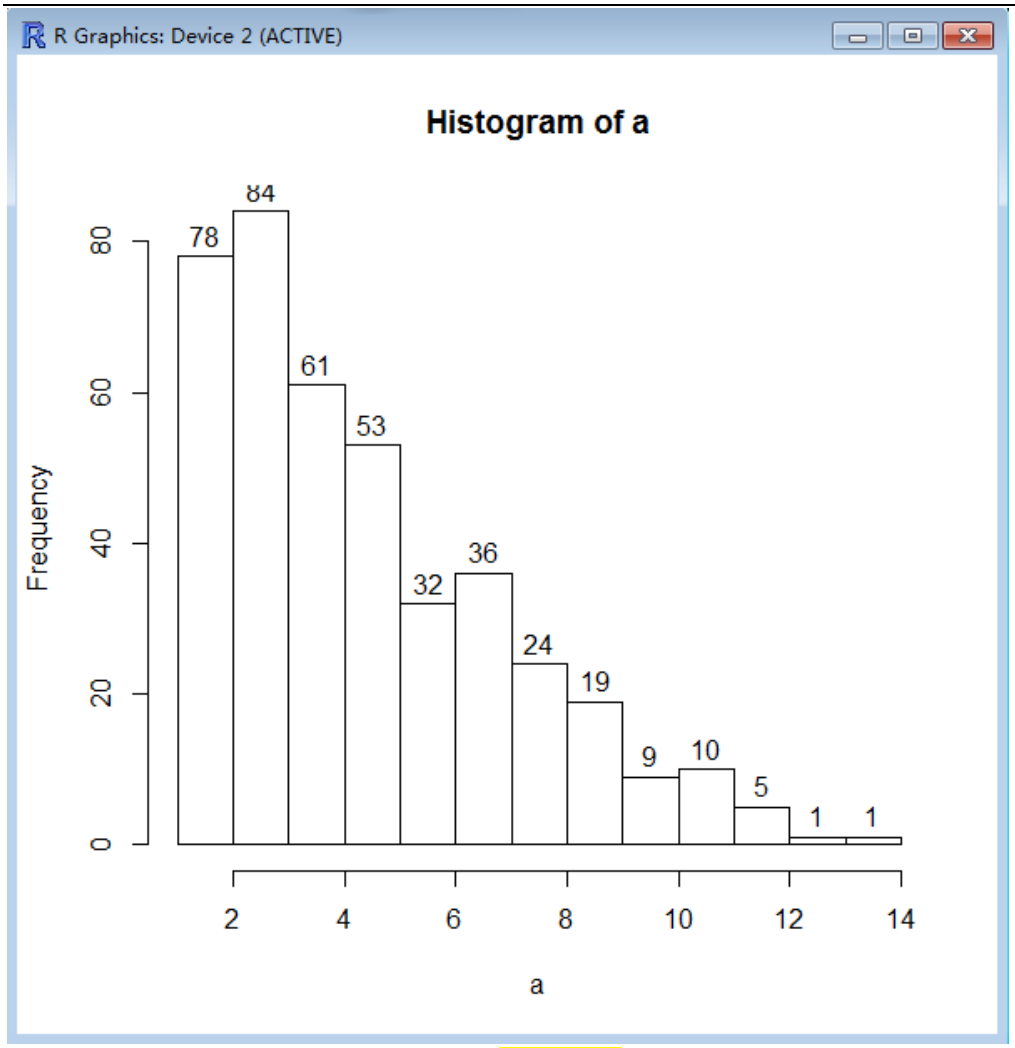

Figure 17

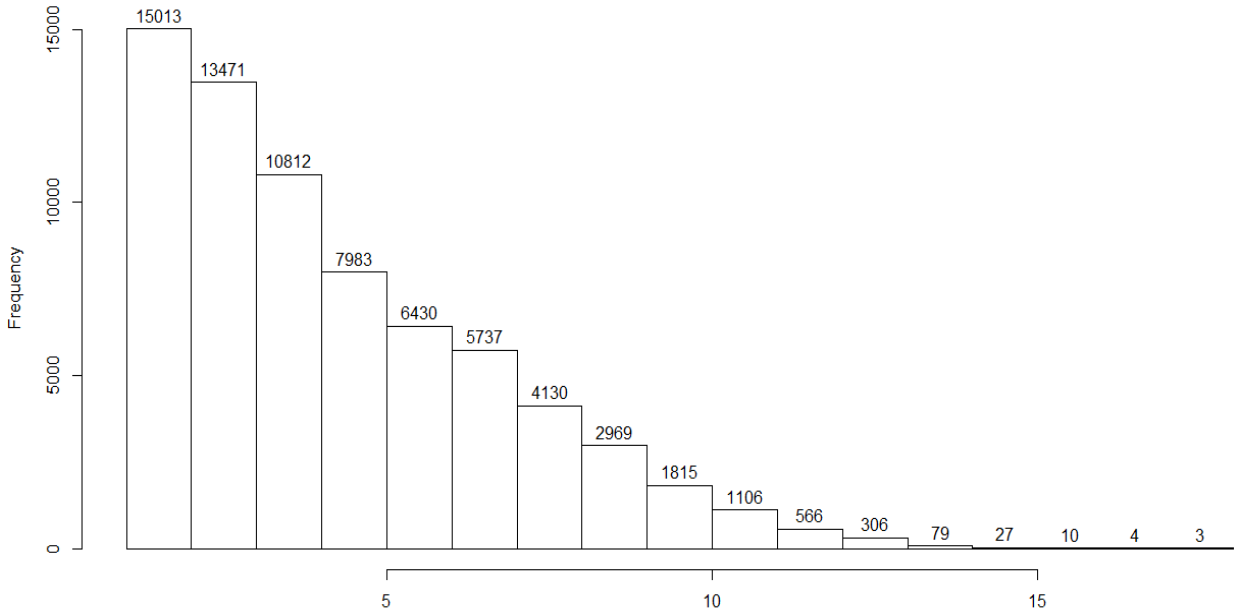

Figure 18

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

```
type="taxonomic_classifier"
```

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：

```
#input:filename
#calls:functions in XML package
#output:the categories of a file(vector)
question5.get_cat<- function(filename){
```

Figure 20

```
#input:NULL
#calls:question5.get_cat
#output:draw a picture of the number of documents in each category

question5.category<- function(){
```

Figure 21

Command.R 中的指令：

```
#Visualize the number of documents in each category

ans5 = question5.category()
save(ans5,file = "./data/Q5.Rdata")
```

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）

```
> ans5
b
                                                                                  Top/Classifieds/Job Market/Job Categories/Art and Design
                                                                                                                                        29
                                                                       Top/Classifieds/Job Market/Job Categories/Banking, Finance and Insurance
                                                                                                                                         1
                                                                    Top/Classifieds/Job Market/Job Categories/Government, Philanthropy and NGO
                                                                                                                                         1
                                                                  Top/Classifieds/Job Market/Job Categories/Hospitality, Restaurant and Travel
                                                                                                                                         1
                                                                                  Top/Classifieds/Job Market/Job Categories/Legal
                                                                                                                                         1
                                                              Top/Classifieds/Job Market/Job Categories/Manufacturing, Operations and Logistics
                                                                                                                                         1
                                                                   Top/Classifieds/Job Market/Job Categories/Marketing, Advertising and PR
                                                                                                                                         2
                                                                Top/Classifieds/Job Market/Job Categories/Media, Entertainment and Publishing
                                                                                                                                         3
                                                                      Top/Classifieds/Job Market/Job Categories/Music, Theater and Dance
                                                                                                                                        23
                                                          Top/Classifieds/Job Market/Job Categories/Technology, Telecommunications and Internet
                                                                                                                                         2
```

Figure 23

见 category.pdf

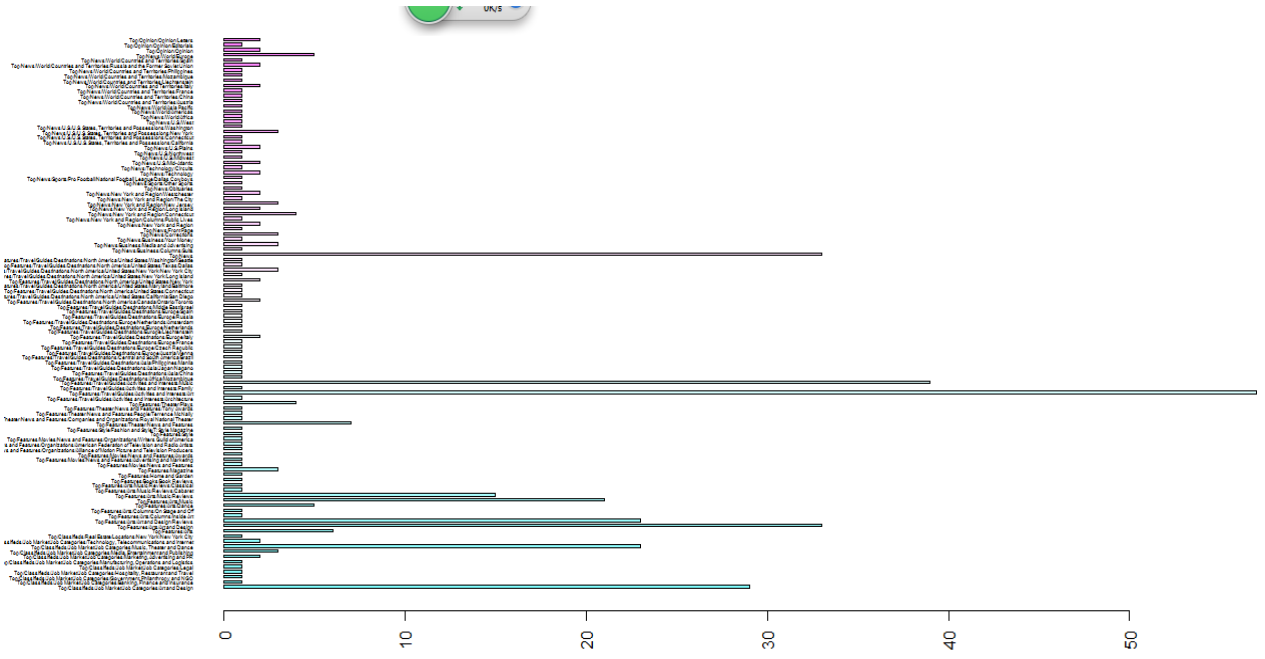

Figure 24

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

```
<meta content="8" name="publication_month"/>
```

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：

```
#input:NULL
#calls:question6.get_month
#output:draw a picture of the number of documents in each month

question6.monHist <- function(){
```

Figure 26

```
#input:filename
#calls:functions in XML package
#output:month of a document
question6.get_month<- function(filename){
```

Figure 27

Command.R 中的指令：

```
#Visualize the number of documents in each month

ans6 = question6.monHist()
save(ans6,file = "./data/Q6.Rdata")
```

Figure 28

四、【结果展示】

数据保存在 data/Q6.Rdata 中
如下图所示：

```
> ans6
  [1] 12 12 10 10  9  5  1  9  4  3  3  3  1 11 10  9  7  6  5  5  1  7  5  4
 [25]  3  4  4  3  3 12  7  7  6  6  3  3  3  2  1  9  9  7  7  6  6  6  3  2
 [49]  2 12  7  5  2  1  1 12  5  3  1  8  2 11 10  9  9  9  7  6  4  3 12  8
 [73]  6  4  4  3 12 11  4  4 11 10 10 10  5  1 10  6  4  7  7  4  3  3 11  6
 [97]  9  8  5  9  8  3
```

Figure 29

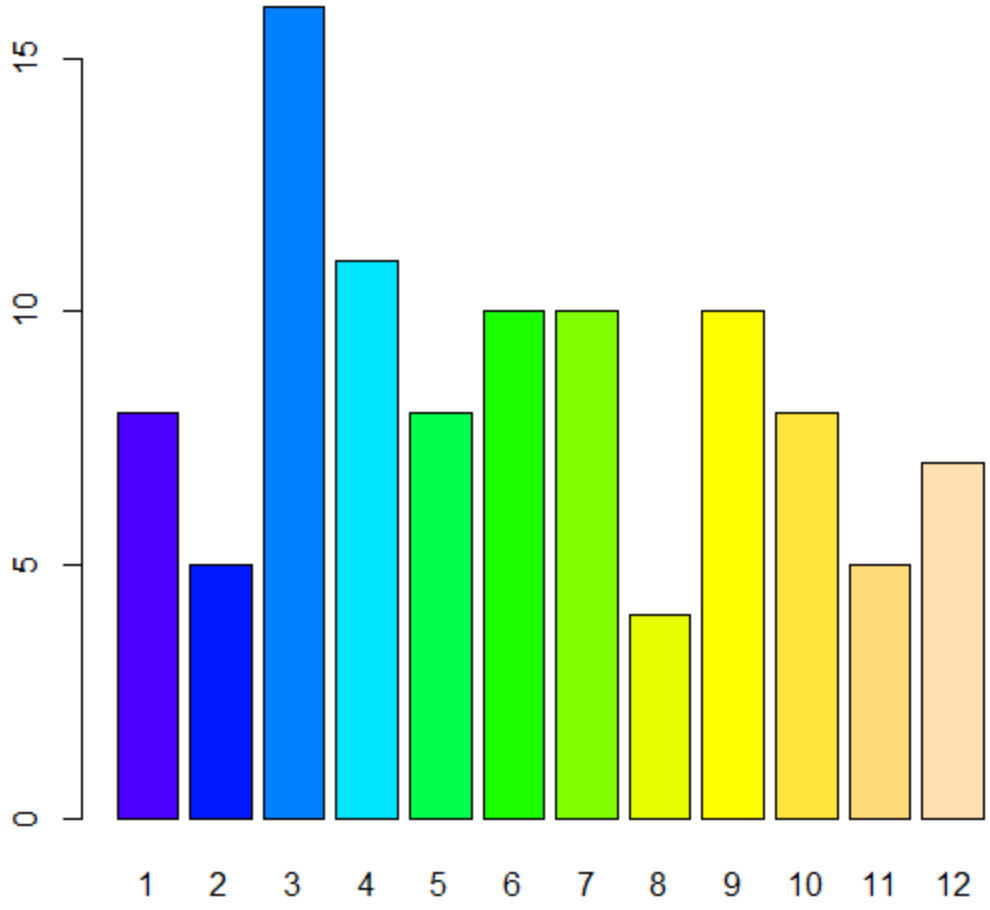

Figure 30

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

```
<dateline>ERCOLANO, Italy</dateline>
```

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：

```
#input:filename
#calls:functions in XML package
#output:month of a document

question7.dateline<- function(filename){
```

Figure 32

```
#input:NULL
#calls:question7.dateline
#output:draw a picture of the number of documents in each location

question7.location<- function(){
```

Figure 33

Command.R 中的指令：

```
#Visualize the number of documents in each location

ans7 = question7.location()
save(ans7,file = "./data/Q7.Rdata")
```

Figure 34

四、【结果展示】

数据保存在 data/Q7.Rdata 中
如下图所示：

```
> ans7
 [1] "WASHINGTON"       "KANSAS CITY"      "ERCOLANO"
 [4] "JERSEY CITY"      "PURCHASE"         "MANILA"
 [7] "EDINBURGH"        "YORKTOWN HEIGHTS" "MAPUTO"
[10] "HOUSTON"          "TORONTO"          "UNIONDALE"
[13] "VIENNA"
```

Figure 35

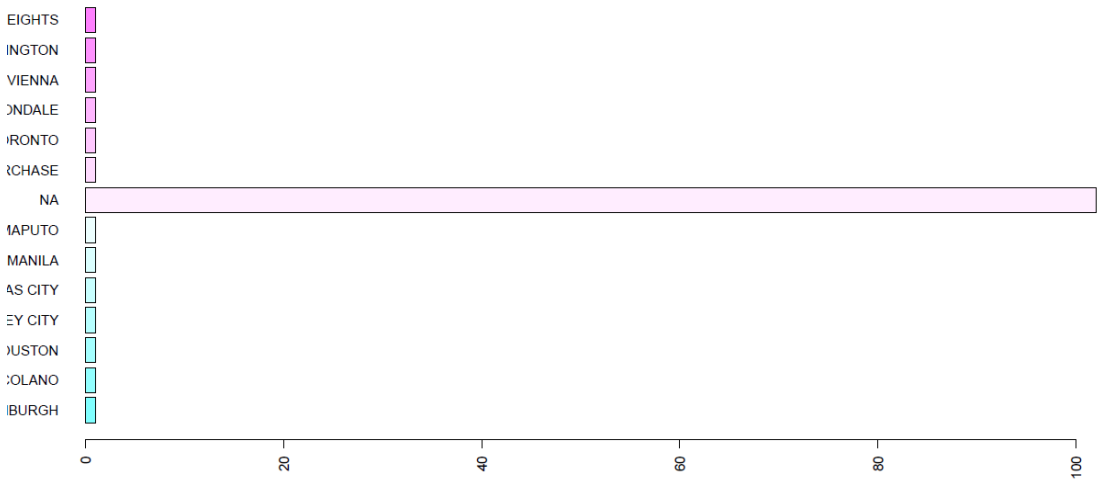

Figure 36

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。**

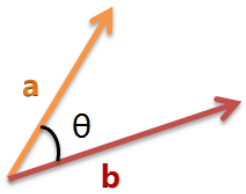

Figure 37

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：

$$\cos\theta = \frac{\sum_{i=1}^{n}(A_i \times B_i)}{\sqrt{\sum_{i=1}^{n}(A_i)^2} \times \sqrt{\sum_{i=1}^{n}(B_i)^2}} = \frac{A \cdot B}{|A| \times |B|}$$

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：

```
#input:x,y as vector
#calls:NULL
#output:the scalar product of two vector

scalar.product <- function(x,y = x){
```

Figure 39

```
#input:bag-of-words data-frame calculated in question3()
#calls:scalar.product
#output:distance matrices

similarity.cosine <- function(x){
```

Figure 40

Command.R 中的指令：

```
#Create distance matrices from this data frame for the cosine "distance"

Tans1 = similarity.cosine(ans3)
save(Tans1,file = "./data/2_Q1.Rdata")
```

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：

```
Tans1
          [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]
[1,] 1.0000000 0.8461390 0.7311328 0.7587374 0.7485679 0.6802264 0.8117710
[2,] 0.8461390 1.0000000 0.7949648 0.8746553 0.8206206 0.7068793 0.8954954
[3,] 0.7311328 0.7949648 1.0000000 0.8052214 0.7162799 0.7090872 0.7549808
[4,] 0.7587374 0.8746553 0.8052214 1.0000000 0.7616625 0.6778705 0.7894583
[5,] 0.7485679 0.8206206 0.7162799 0.7616625 1.0000000 0.6524527 0.7983606
[6,] 0.6802264 0.7068793 0.7090872 0.6778705 0.6524527 1.0000000 0.7031958
```

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：

```
#input:NULL
#calls:question5.get_cat
#output:Mapping matrix
classifier.category <- function(){
```

Figure 43

```
#input:matrice getted from classifier.category
#calls:NULL
#output:categories distribution

refresh_vector <- function(dd){
```

Figure 44

```
#input:number of files in a category
#calls:NULL
#output:sum
sc.same.category <- function(xx,sc){
```

Figure 45

```
#input:matrice getted from refresh_vector
#calls:similarity.cosine and sc.same.category
#output:the average distance between stories in the same category
same.average.distance<- function(dd){
```

Figure 46

```
#input:sc is the cosine product matrices
#calls:NULL
#output:sum of distances

sc.differ.category <- function(x,y,sc){
```

Figure 47

```
#input:matrice getted from refresh_vector
#calls:NULL
#output:vector for catergory[col]
frame.col <- function(dd,col){
```

Figure 48

```
#input:matrice getted from refresh_vector
#calls:similarity.cosine and sc.differ.category
#output:the average distance between stories in the different category
differ.average.distance <- function(dd){
```

Figure 49

Command.R 中的指令：

```
a = classifier.category()
b = refresh_vector(a)
Tans2_1 = same.average.distance(b)
Tans2_2 = differ.average.distance(b)
save(Tans2_1,Tans2_2,file = "./data/2_Q2.Rdata")
```

Figure 50

四、【结果展示】

数据保存在 data/2_Q2.Rdata 中
如下图所示：

```
> Tans2_1
[1] 0.705089
> Tans2_2
[1] 0.701802
```

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。

$$\text{词频(TF)} = \frac{\text{某个词在文章中的出现次数}}{\text{文章的总词数}}$$

Figure 52

第二步，计算逆文档频率。

$$\text{逆文档频率(IDF)} = \log\left(\frac{\text{语料库的文档总数}}{\text{包含该词的文档数} + 1}\right)$$

Figure 53

第三步，计算 TF-IDF。

$$\text{TF} - \text{IDF} = \text{词频(TF)} \times \text{逆文档频率（IDF）}$$

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。

```
bb <- TFIDF.weight(bb) * bb

query.new <- TFIDF.weight(query.new) * query.new
```

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：

```
#input:data-frame of documents
#calls:NULL
#output:matrices of TFIDF

TFIDF.weight <- function(aa){
```

Figure 55

```
#input:names of data-frame getted from question3() and word vector of the query
#calls:NULL
#output:new vector of the query without useless words
delete <-function(x,name.dd){
```

Figure 56

```
#input:names of data-frame getted from question3() and a query
#calls:delete,make.BoW.frame in "01.R" and TFIDF.weight
#output:list of best matches documents and nearest distance
question2_3 <- function(dd,query){
```

Figure 57

```
#This function written for converting a row number to filename
#input:a row number
#calls:NULL
#output:filename
convert_to_filename <- function(a)
```

Figure 58

Command.R 中的指令：

```
#find the document which best matches a given query string.

query2 <- question1("0023931.xml")
query3 <- question1("0068412.xml")
query1 <- "An old woman had a cat. The cat was very old; she could not run quickly, and she could not bite, because she was so ol
Tans2_3_1 <- question2_3(ans3,query1) #neariest xml number and distance
Tans2_3_2 <- question2_3(ans3,query2)
Tans2_3_3 <- question2_3(ans3,query3)

aa = convert_to_filename(Tans2_3_1) #convert xml number to filename
bb = convert_to_filename(Tans2_3_2)
cc = convert_to_filename(Tans2_3_3)

save(Tans2_3_1,Tans2_3_2,Tans2_3_3,aa,bb,cc,file = "./data/2_Q3.Rdata")
```

Figure 59

四、【结果展示】

数据保存在 data/2_Q3.Rdata 中
如下图所示：

```
> Tans2_3_1
$which
[1] 29

$dist
[1] 1.402979

> Tans2_3_2
$which
[1] 1

$dist
[1] 0.8105651

> Tans2_3_3
$which
[1] 2

$dist
[1] 16.09267
```

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。