R 机器学习

第 18 讲　文本分析

张敬信

2022 年 11 月 29 日

哈尔滨商业大学

一. 文本分析简介

当今社会，每天都会产生海量的文本信息，如新闻、评论、聊天、报告、文档等，随之而来出现了各种基于定量和计算方法的文本挖掘和文本分析技术，目前已广泛应用于政治、政策等社会科学，以及金融、媒体和通信等领域。

文本分析，是用统计方法和机器学习对文本进行分析以获得洞察力，以及如何使用图形方法展示结果。

文本分析，是将自然语言文件形式的非结构化原始数据转化为可系统分析的结构化数据，并使用特定的定量分析方法进行分析，包括**描述性分析、关键词分析、主题分析、度量与标准化、文本聚类、文本比较、情感分析，以及因果推断、预测模型**等。

另外，**自然语言处理（NLP）**是目前的热门领域，通过语言生成模型和深度学习研究如何让计算机读懂人类语言。

quanteda 包用于管理和分析文本数据，代表了最新的 R 文本分析生态。

该包从底层开始重新设计了文本处理过程，在语法与性能上表现出色：

- 内部使用 stringi 包作为字符处理工具
- 内部基于 data.table 包与 Matrix 包
- 统一的语法结构，稳定一致的工作流程

《Text Analysis Using R》是 quanteda 官方出品的新书（正在写）。

《Text Mining with R: A Tidy Approach》，tidy 风格的文本挖掘。

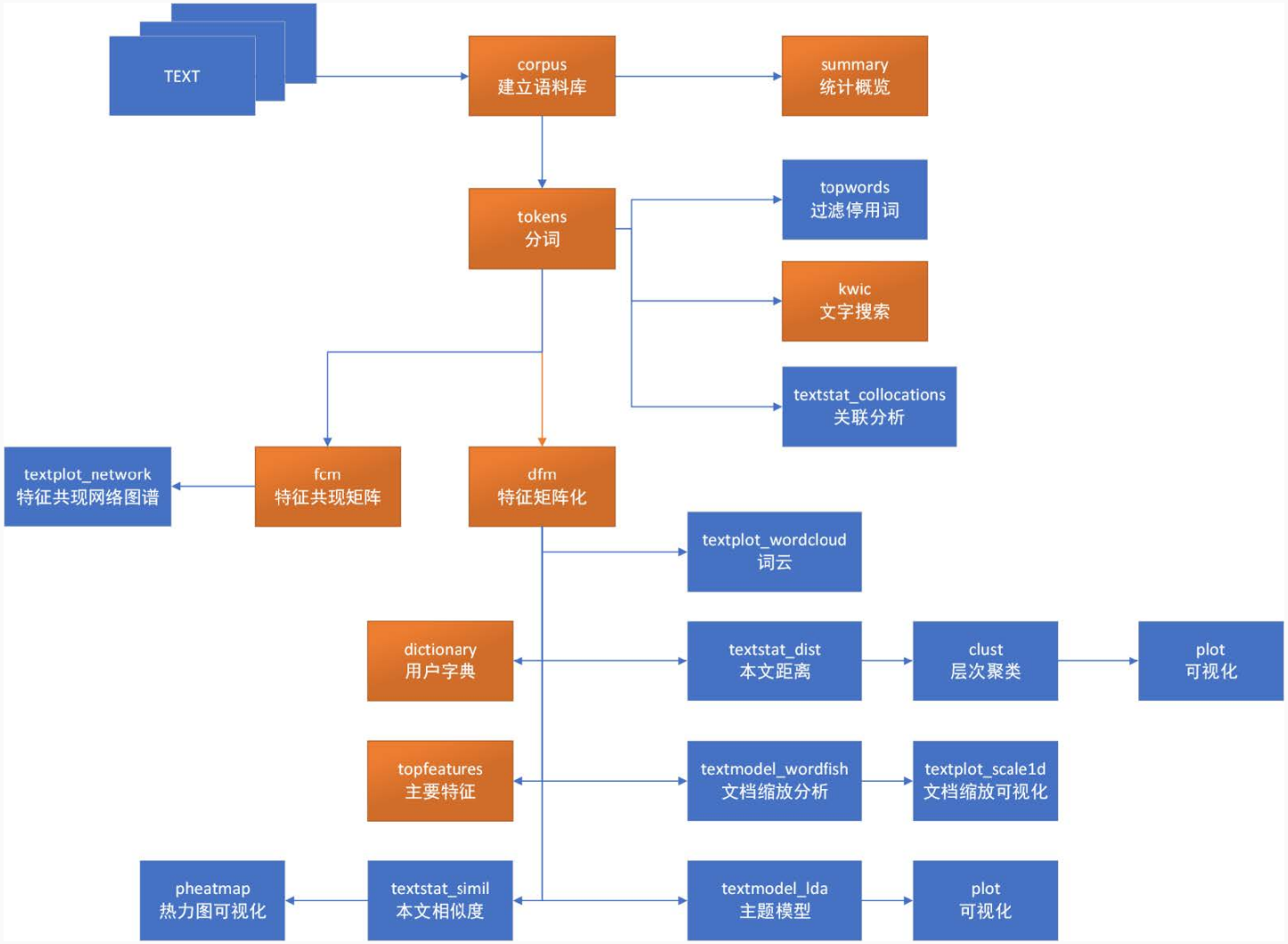


图 1: quanteda 文本处理流程

一. 语料

每个文本分析项目都从一组文本开始，这些文本称为**语料**。

语料通常为共同目的而收集的文件合集，每个语料通常有三个构成元素：

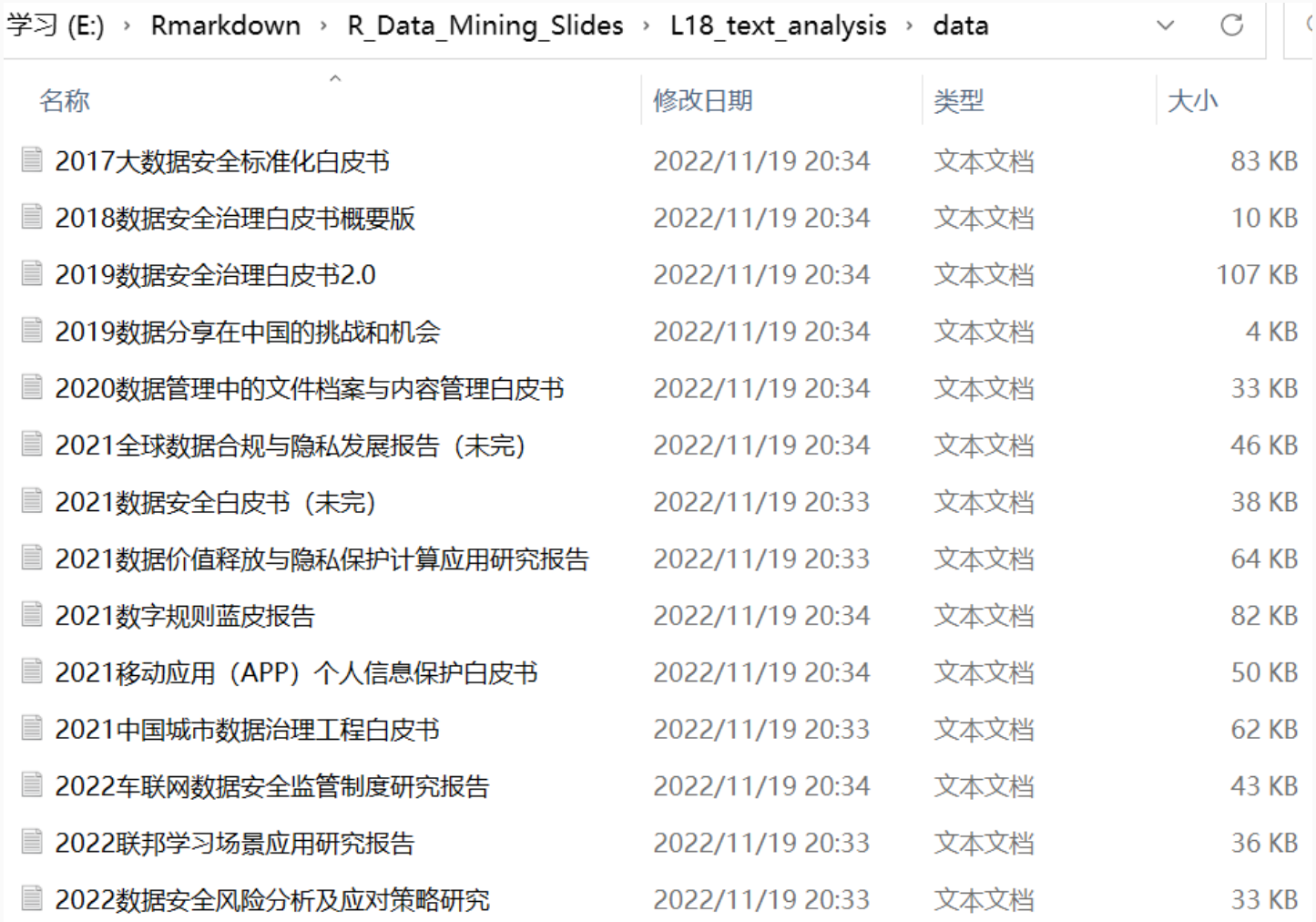
- 文本数据的文档
- 文档级变量（跟各文档相关联的变量）
- 关于整个语料的元数据（描述，作者，来源等信息）

可以从各种可用的来源创建一个语料：

- 字符向量：每个元素都是一个文档
- 由文档的字符向量和额外的文档级变量构成的数据框
- 由 `tm` 包创建的 `VCorpus` 或 `SimpleCorpus` 类对象

1. 准备数据

现有来自和鲸社区的**数据安全文本数据**：



名称	修改日期	类型	大小
2017大数据安全标准化白皮书	2022/11/19 20:34	文本文档	83 KB
2018数据安全治理白皮书概要版	2022/11/19 20:34	文本文档	10 KB
2019数据安全治理白皮书2.0	2022/11/19 20:34	文本文档	107 KB
2019数据分享在中国的挑战和机会	2022/11/19 20:34	文本文档	4 KB
2020数据管理中的文件档案与内容管理白皮书	2022/11/19 20:34	文本文档	33 KB
2021全球数据合规与隐私发展报告（未完）	2022/11/19 20:34	文本文档	46 KB
2021数据安全白皮书（未完）	2022/11/19 20:33	文本文档	38 KB
2021数据价值释放与隐私保护计算应用研究报告	2022/11/19 20:33	文本文档	64 KB
2021数字规则蓝皮报告	2022/11/19 20:34	文本文档	82 KB
2021移动应用（APP）个人信息保护白皮书	2022/11/19 20:34	文本文档	50 KB
2021中国城市数据治理工程白皮书	2022/11/19 20:33	文本文档	62 KB
2022车联网数据安全监管制度研究报告	2022/11/19 20:34	文本文档	43 KB
2022联邦学习场景应用研究报告	2022/11/19 20:33	文本文档	36 KB
2022数据安全风险分析及应对策略研究	2022/11/19 20:33	文本文档	33 KB

图 2：将作为语料的原始文本数据

readtext 包的 readtext() 函数是专门用来读取文本数据的，支持各种常见的文本文件类型：txt、json、csv/tab/tsv、html/xml、pdf、doc/docx、rtf 等，甚至是文件路径、网址、压缩包。

```
library(tidyverse)
library(quanteda)
library(readtext)
dat = readtext("data/*.txt", encoding = "UTF-8")
```

	doc_id	text
1	2017大数据安全标准化白皮书.txt	随着社会信息化和网络化的发展，数据爆炸式增长，大数据...
2	2018数据安全治理白皮书概要版.txt	数据治理或数据安全概念对于大多数IT和安全从业者来说,认...
3	2019数据安全治理白皮书2.0.txt	数据治理或者数据安全概念，对于大多数IT和安全从业者来...
4	2019数据分享在中国的挑战和机会.txt	总的来说，在中国有很多研究人员正在创建DMP (93%)，但...
5	2020数据管理中的文件档案与内容管理白皮书.txt	随着大数据、人工智能等信息技术的快速发展，数据的价值...
6	2021全球数据合规与隐私发展报告（未完）.txt	全球迈入数字经济时代，数据成为驱动各国经济社会创新发...
7	2021数据安全白皮书（未完）.txt	2019年，全球数字经济规模达到31.8万亿美元。 从单- -国家...
8	2021数据价值释放与隐私保护计算应用研究报告.txt	2020年10月，《中华人民共和国国民经济和社会发展第十四...
9	2021数字规则蓝皮报告.txt	数字化发展引发各领域、各行业生产模式、商业模式、管理...
10	2021移动应用（APP）个人信息保护白皮书.txt	随着移动通信技术的飞速发展，移动应用已渗透到人们生活...
11	2021中国城市数据治理工程白皮书.txt	现代数字城市建设是推进国家治理体系和治理能力现代化的...
12	2022车联网数据安全监管制度研究报告.txt	随着现代信息技术的不断发展，汽车行业已经进入新能源数...
13	2022联邦学习场景应用研究报告.txt	数据作为数字经济和信息社会的核心资源，被认为是继土地...
14	2022数据安全风险分析及应对策略研究.txt	数字化时代，数据已成为数字经济发展的核心生产要素，20...

图 3: 将用来创建语料的数据框（部分）

dat 是数据框，14 个 txt 文件对应 14 个观测，doc_id 列为文档名字，text 列为文本数据，每个值是一个 txt 文档内容构成的字符串。

数据框格式，便于：

- 对文本内容（字符串）做预处理（清洗），比如，有的文档段内有不必要的换行，"。\n" 是正常分段，而" 非句号\n" 应该删除掉"\n"，以避免词语不恰当地断开，影响后续的分词操作；
- 创建文档级变量，比如每个文档的作者、作者身份、日期等。

```
dat = dat %>%
  mutate(text = str_replace_all(text, "(?<=[^。])\n", ""),
         year = str_extract(doc_id, "^\\d{4}"))
```

2. 创建语料

用 corpus() 将数据框创建为语料，text_field 指定从 text 列创建，其余列除了 doc_id 将自动作为文档级变量：

```
corp = corpus(dat, text_field = "text")
head(corp, 2)
#> Corpus consisting of 2 documents and 1 docvar.
#> 2017 大数据安全标准化白皮书.txt :
#> " 随着社会信息化和网络化的发展，数据爆炸式增长，大数据时代
#>
#> 2018 数据安全治理白皮书概要版.txt :
#> " 数据治理或数据安全概念对于大多数 IT 和安全从业者来说，认
```

注：将语料转化为数据框用 convert(corp, to = "data.frame").

3. 文档级变量与语料子集

文档级变量对于许多文本分析项目来说是至关重要的，比如想要比较不同年份数据安全的重视程度，就需要一个文档年份的文档级变量。

可用 `corp$newcol = ...` 创建新的文档级变量，用 `docvars()` 返回文档级变量构成的数据框。

借助文档级变量构造筛选条件，可以对语料提取子集：

```
corpus_subset(corp, year == 2022)
```

还有对语料随机抽样提取子集：

```
corpus_sample(corp, size = 5, replace = FALSE)
# by 设置分组抽样，prob 指定抽样概率
```

4. 改变文档单位

语料是以文档为单位组织文本数据，上述方式创建语料是一个文件作为一个文档。当然也可以重新分割或按组合并，以改变文档单位。比如将评论数据按每周/天分割，将按节的小说合并成章。

- `corpus_segment(x, pattern)`：按正则表达式分割语料；
- `corpus_reshape(x, to = c(sentences", "paragraphs", "documents"))`：重塑为不同聚合单位；
- `corpus_group(x, groups)`：按文档级分组变量，向上合并文本数据。

二. 分词

将语料中的文本按词的边界分割成词元（词或句子），称为**分词**。

用 `tokens()` 对语料做分词，参数 `what` 设置分词方式：`"word"`（默认），`"sentence"`，`"character"`（字符），还有参数 `remove_punct/remove_sympols/remove_numbers` 等设置是否删除标点符号/特殊符号/数字等。

若按段落分词，可借助 `tidytext::tokenize_paragraphs()`.

`tokens()` 内部基于 `stringi` 包，也支持中文语料分词。但是，更建议用效果更好的 `jieba` 分词。

1. quanteda 自动分词

```
toks = tokens(corp, what = "word",
              remove_punct = TRUE, remove_symbols = TRUE)
head(toks, 2)
#> Tokens consisting of 2 documents and 1 docvar.
#> 2017 大数据安全标准化白皮书.txt :
#>  [1] "随着" "社会" "信息" "化"   "和"   "网络" "化
#> [11] "爆炸" "式"
#> [ ... and 15,196 more ]
#>
#> 2018 数据安全治理白皮书概要版.txt :
#>  [1] "数据" "治理" "或"   "数据" "安全" "概念" "对
#> [11] "和"   "安全"
#> [ ... and 1,709 more ]
```

2. 手动 jieba 分词

- 先用 worker() 初始化分词引擎，可以提供用户词典（1 个词占 1 行的文本文档），通常是下载搜狗输入法的分类细胞词库，再将一些目测分错的词放进去；
- 对原数据的 txt 列，依次做 jieba 分词，并用 doc_id 命名得到命名列表，再交给 tokens().

```
# jieba 分词
library(jiebaR)
wk = worker(user = "user.dict.utf8")
toks = map(dat$text, ~ segment(.x, wk)) %>%
  set_names(dat$doc_id) %>%
  tokens()
```

可见，分词效果更好：

```
head(toks, 2)
#> Tokens consisting of 2 documents.
#> 2017 大数据安全标准化白皮书.txt :
#>  [1] "随着"   "社会"   "信息化" "和"     "网络化" "
#>  [9] "爆炸式" "增长"   "大"     "数据"
#> [ ... and 13,521 more ]
#>
#> 2018 数据安全治理白皮书概要版.txt :
#>  [1] "数据"     "治理"     "或"       "数据安全" "概
#>  [7] "大多数"   "IT"       "和"       "安全"     "从
#> [ ... and 1,534 more ]
```

但是，这样手动创建的分词对象 toks，还没有文档级变量。若需要，用同样的方法创建即可：

```
toks$year = str_extract(dat$doc_id, "^\\d{4}")
names(toks) = str_c(" 文档", 1:14, ".txt")
map_int(toks, length) |> `names<-`(NULL)
#>  [1] 13533  1546 18099   635  5538  7672  5981 10582 12
#> [13]  5937  5286
```

- 剔除标点符号、数字

```
toks = tokens(toks, remove_punct = TRUE,
              remove_numbers = TRUE) %>%
  tokens_tolower()     # 英文通常需要转化小写
```

3. 剔除停用词

停用词是任何语言中出现频率很高，但没有多大意义，只是用来支持句子结构的虚词，比如：的，了，呢，并且，the，……

停用词需要剔除掉，一是避免淹没主要词，二是大大降低文档特征矩阵的维度。

```
zh_stop = stopwords("zh", source = "misc")    # 字符向量，64
toks = tokens_remove(toks, zh_stop)
map_int(toks, length) |> `names<-`(NULL)
#>  [1]  9806  1057 12455   409  4103  5636  4444  7903 10
#> [13]  4202  4067
```

注：tokens_select() 可借助正则表达式正选。

也可以读取一些专用的停用词表（百度、哈工大），或者自己选出来若干停用词，继续剔除：

```
baidu = readLines(" 百度停用词.txt") %>%
  str_split(" ") |> unlist()
toks = tokens_remove(toks, baidu)
mywords = c(" 一个", " 认为", " 已经")
toks = tokens_remove(toks, mywords)
map_int(toks, length) |> `names<-`(NULL)
#>  [1] 8022  858 9949  322 3427 4641 3658 6571 8545 4834 (
```

4. 关键词查询

分词对象仍然保留了词的顺序，可以用来查询关键词的出现上下文：

```
kwic(toks, pattern = c(" 信息安全"), window = 2) %>%
  head(4)
#> Keyword-in-context with 4 matches.
#>    [文档 1.txt, 192]       工作 全国 / 信息安全 / 标准化 技
#>  [文档 1.txt, 1251] 必要措施 确保 / 信息安全 / 业务 活动
#>  [文档 1.txt, 1290] 个人信息 规则 / 信息安全 / 保障 措施
#>  [文档 1.txt, 1311] 必要措施 确保 / 信息安全 / 消费者 个
```

5. 合并词组

英文中有多个单词构成的词组，比如 European Union, New York 等，中文也有并列名词表义，比如就业质量、劳动经济学等。

这样的词组，应该合并成一个词。可以放在一个文本文档中，处理成特殊格式：

```
words = read_lines("English.txt") %>%
  map(~ str_split(.x, " ") |> unlist())
words
#> [[1]]
#> [1] "United"  "Kingdom"
#>
#> [[2]]
#> [1] "European" "Union"
```

再传递给 tokens_compound()，对词元做合并：用下划线连接成一个词

```
txt = "The United Kingdom is leaving the European Union."
tk = tokens(txt, remove_punct = TRUE)
tokens_compound(tk, words)
#> Tokens consisting of 1 document.
#> text1 :
#> [1] "The"            "United_Kingdom" "is"
#> [5] "the"            "European_Union"
```

另外，还有 tokens_ngrams() 生成 n 单词序列，常用的有 2-gram 和 3-gram.

6. 英文词干化

同一含义的英文单词经常有多种形式，比如：win, winning, wins 等，有必要转化成同一个词，这称为**词干化**。

```
txt = c(one = "win winning wins won winner",
        two = "taxing taxes taxed my tax return")
tk = tokens(txt)
tokens_wordstem(tk)
#> Tokens consisting of 2 documents.
#> one :
#> [1] "win"    "win"    "win"    "won"    "winner"
#>
#> two :
#> [1] "tax"    "tax"    "tax"    "my"     "tax"    "retur
```

三. 文档特征矩阵

文档特征矩阵，一个文档占一行，一个词元是一个特征（作为列名），单元格的值为词元在文档出现的频数。

用 dfm() 将分词对象创建为文档特征矩阵：

```
doc.dfm = dfm(toks) %>%
  dfm_trim(min_docfreq = 2)
ndoc(doc.dfm)
#> [1] 14
nfeat(doc.dfm)
#> [1] 3092
featnames(doc.dfm) %>% head(10)
#>  [1] "社会"   "信息化" "网络化" "发展"   "数据"   "
#>  [9] "到来"   "世纪"
```

```
topfeatures(doc.dfm, 20)     # 出现频数最高的 20 个特征
#>       数据   数据安全       保护       技术       管理       隐私   个人
#>       4426        863        673        563        559        551
#>       企业       发展       治理       用户       业务       风险
#>        446        437        407        369        357        335
#>       相关       过程       平台       能力
#>        279        277        275        272
```

`textstat_frequency()` 提供了更高级的统计功能，返回数据框方便后续分析。

另外，有时候只考虑词频不一定恰当，有些词虽然出现的频率高，但如果天生就在很多文档中普遍存在，那么其重要性可能就没那么高。这就需要考虑逆文档频率（衡量一个词在一组文档中的流行程度）：

$$idf(t, D) = \ln \frac{|D|}{|\{d \in D : t \in d\}|}$$

然后，用 $tf(t, d)$ 表示文档 d 中词项 t 的词频，则词频-逆文档频率定义为：

$$tf\_idf(t, d, D) = tf(t, d) \cdot idf(t, D)$$

用 `dfm_tfidf()` 转化词频-逆文档频率，相当于是对特征频数根据文档数做加权。另一种方法是用 `dfm_weight()` 对特征频数加权。

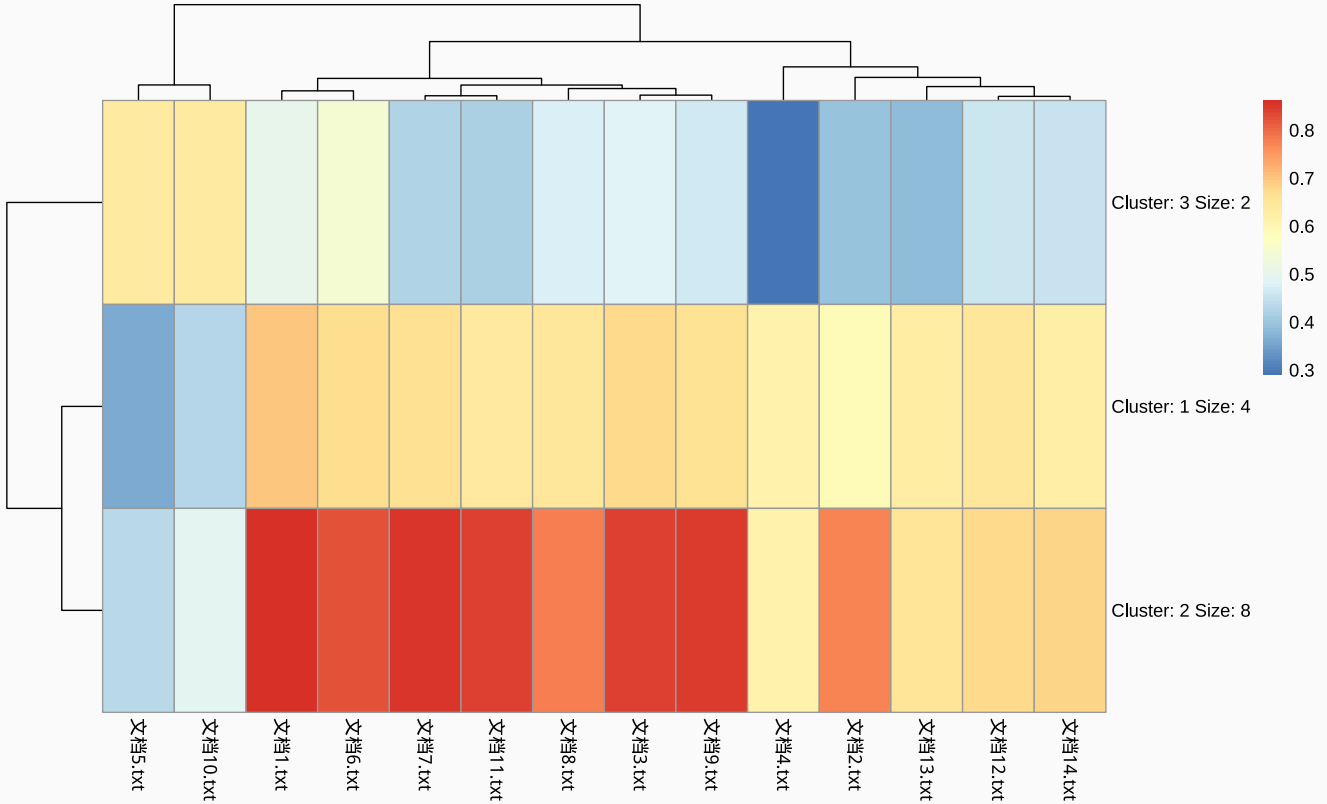
1. 词云图

```r
library(quanteda.textplots)
set.seed(1)
textplot_wordcloud(doc.dfm, min_count = 100,
  random_order = FALSE, rotation = 0.25, min_size = 1,
  max_size=4, color=RColorBrewer::brewer.pal(8, "Dark2"))
```

2. 文本相似

```
library(quanteda.textstats)
library(pheatmap)
simi = textstat_simil(doc.dfm, margin = "documents",
                      method = "cosine")
```

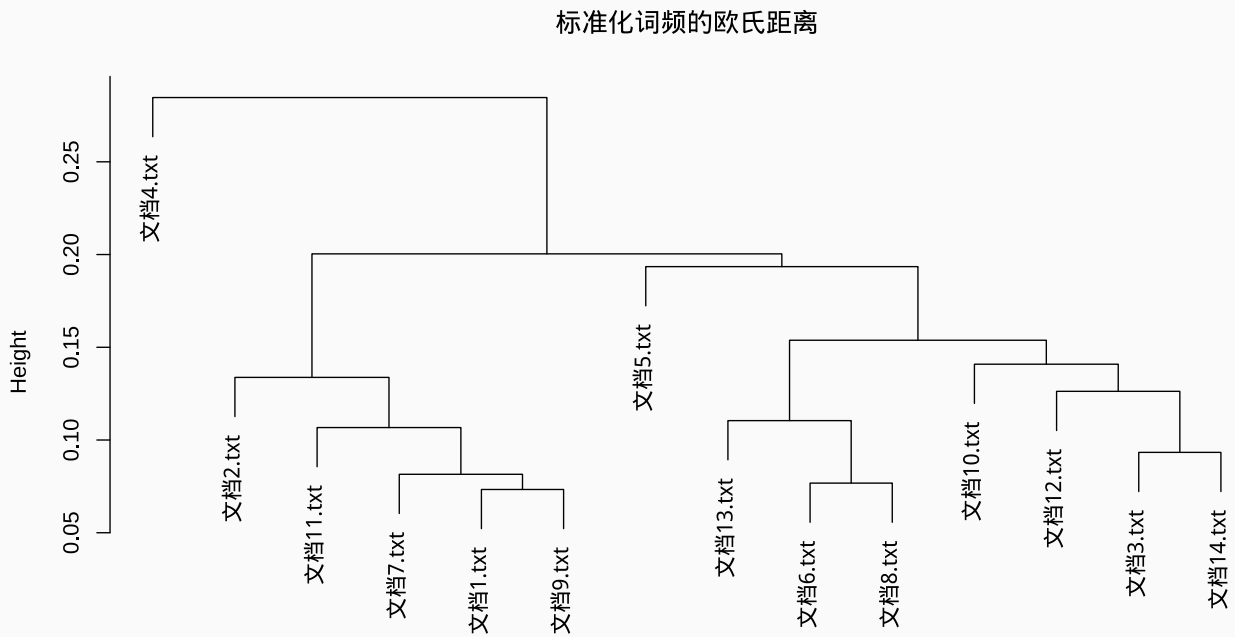
```
pheatmap(as.matrix(simi), kmeans_k = 3)
```



3. 文本聚类

```
dfm_tmp = dfm_trim(doc.dfm, min_termfreq = 50,
                   min_docfreq = 2)
# 分层聚类 - 在归一化 dfm 上计算距离
dist_tmp = textstat_dist(
  dfm_weight(dfm_tmp, scheme = "prop"))
# 聚类分析文本距离
cluster_tmp = hclust(as.dist(dist_tmp))
# 按文档名标注
cluster_tmp$labels = docnames(dfm_tmp)
```

```
# 绘制树状图
plot(cluster_tmp, xlab = "", sub = "",
     main = " 标准化词频的欧氏距离")
```



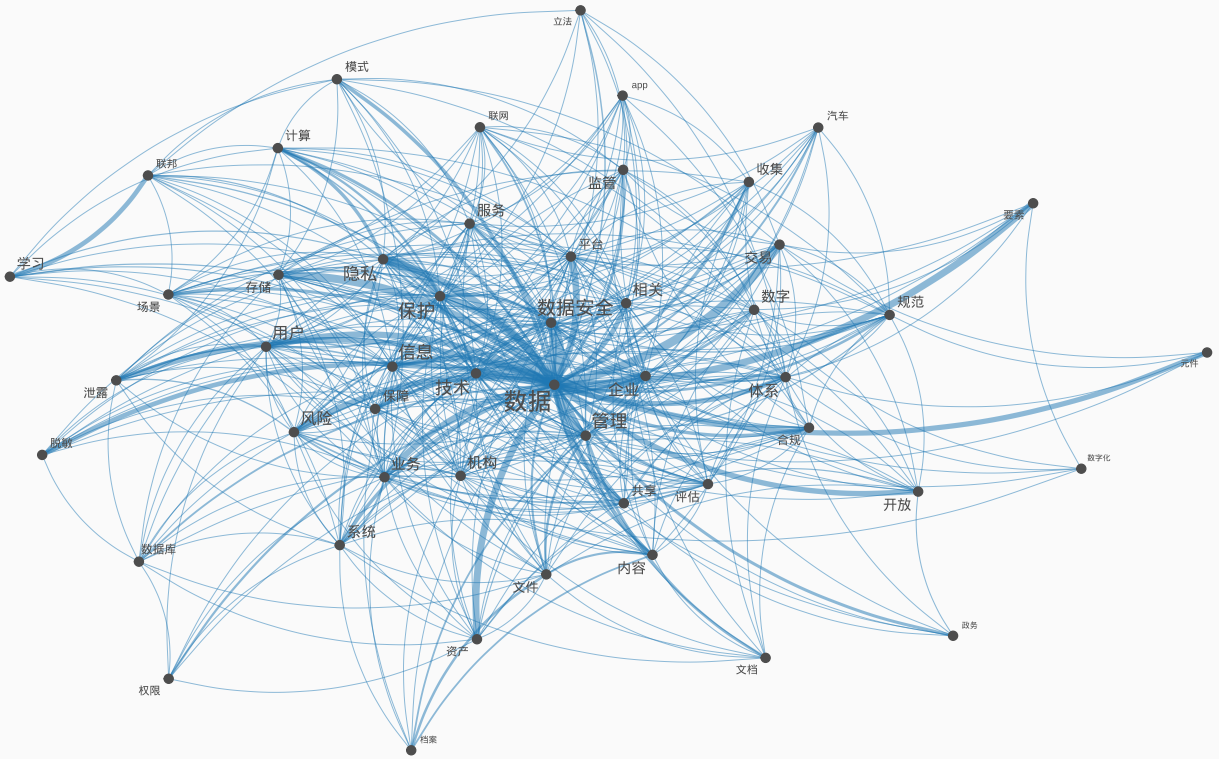
关于文本分类：若文本数据有因变量，比如电影评论的情感得分（“pos”，“neg”），则可以以文档特征矩阵或其嵌入为自变量，做有监督分类。通常的分类算法，如朴素贝叶斯、决策树等都可以使用。

4. 词关联分析

```
textstat_collocations(toks, size = 3, min_count = 10,
                      tolower = TRUE) %>% head()
#>       collocation count count_nested length lambda    z
#> 1 利用 数据 技术    10            0      3   3.39 5.27
#> 2 跨境 数据 流动    21            0      3   3.23 3.49
#> 3 企业 数据 合规    10            0      3   1.83 3.24
#> 4 元件 数据 产品    10            0      3   4.32 2.86
#> 5 政府 数据 开放    11            0      3   2.19 2.81
#> 6 数据 跨境 流动    37            0      3   1.54 2.78
```

5. 特征共现矩阵

```
fcmat = fcm(toks, context = "window", window = 5)
feat = names(topfeatures(fcmat, 50))
fcmat = fcm_select(fcmat, pattern = feat)
textplot_network(fcmat, min_freq = 0.5,
  vertex_labelsize = 0.5 * log(rowSums(fcmat) + 1))
```


立法
模式
app
联网
汽车
计算
联邦
监管
收集
服务
平台
交易
学习
隐私
存储
场景
保护
数据安全
相关
数字
规范
用户
信息
泄露
技术
数据
企业
体系
保障
风险
管理
合规
隐私
业务
机构
数字化
共享
评估
开放
系统
数据库
内容
文件
政务
资产
文档
权限
档案

四. LDA 主题模型

文档特征矩阵需要转换为 topicmodels 格式，再运行 LDA 模型。

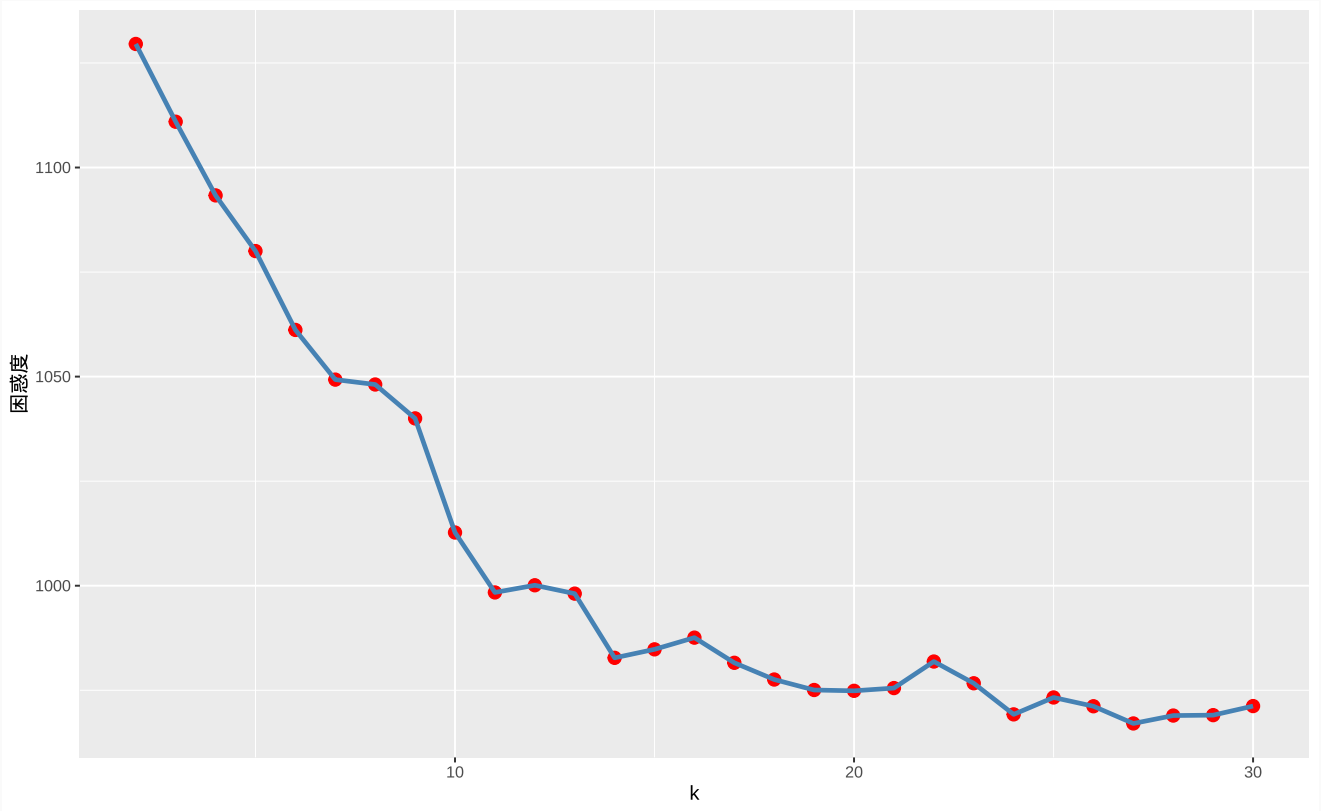
```
library(topicmodels)
dtm = convert(doc.dfm, to = "topicmodels")
```

- 确定最佳主题数

```
set.seed(123)
train = sample(rownames(dtm), nrow(dtm) * 0.8)
dtm_train = dtm[rownames(dtm) %in% train, ]
dtm_test = dtm[!rownames(dtm) %in% train, ]
Calc_Perplexity = function(k, train, test) {
  m = LDA(train, method = "Gibbs", k = k,
          control = list(alpha = 50/k, delta = 0.1))
  perplexity(m, test)
}
```

```
library(furrr)
plan(multisession, workers = 10)
set.seed(1)
rlt = tibble(k = 2:30,
  Perplexity = future_map_dbl(k,
    ~ Calc_Perplexity(.x, dtm_train, dtm_test)))
rlt
#> # A tibble: 29 x 2
#>       k Perplexity
#>   <int>      <dbl>
#> 1     2      1130.
#> 2     3      1111.
#> 3     4      1093.
#> 4     5      1080.
#> 5     6      1061.
#> # ... with 24 more rows
```

```
ggplot(rlt, aes(k, Perplexity)) +
  geom_point(color = "red", size = 3) +
  geom_line(color = "steelblue", size = 1.2) +
  labs(y = " 困惑度")
```



- 确定最佳主题数是 13, 构建最终的主题模型

```
set.seed(123)
mdl = LDA(dtm, method = "Gibbs", k = 13,
          control = list(alpha = 50/13, delta = 0.1))
mdl
#> A LDA_Gibbs topic model with 13 topics.
```

探索 LDA 结果

```
terms(mdl, 10)     # 每个主题最高频的 10 个关键词
#>       Topic 1    Topic 2    Topic 3 Topic 4 Topic 5    T
#>  [1,] "隐私"     "个人信息" "数据"  "数据"  "管理"
#>  [2,] "个人信息" "保护"     "交易"  "要素"  "文件"
#>  [3,] "企业"     "app"      "开放"  "治理"  "内容"
#>  [4,] "保护"     "隐私"     "规则"  "市场"  "文档"
#>  [5,] "合规"     "用户"     "流动"  "资源"  "机构"
#>  [6,] "数据安全" "收集"     "政务"  "流通"  "档案"
#>  [7,] "监管"     "提供"     "数字"  "体系"  "业务"
#>  [8,] "计算"     "合规"     "跨境"  "元件"  "数据管
#>  [9,] "科技"     "监管"     "国家"  "技术"  "结构化
#> [10,] "立法"     "行业"     "收集"  "价值"  "组织"
#>       Topic 8    Topic 9    Topic 10   Topic 11 Topic 12
#>  [1,] "数据安全" "经济"     "汽车"     "隐私"   "数
#>  [2,] "账号"     "存储"     "联网"     "保护"   "技
```

各主题的关键词及概率权重

```
topic_words = function(topic) {
  words = posterior(mdl)$terms[topic, ]
  topwords = head(sort(words, decreasing = TRUE), n = 50)
  rlt = head(topwords, 15)
  tibble(word = names(rlt), prob = rlt)
}
map_dfr(1:13, topic_words, .id = "topic")
#> # A tibble: 195 x 3
#>   topic word       prob
#>   <chr> <chr>     <dbl>
#> 1 1     隐私     0.0401
#> 2 1     个人信息 0.0291
#> 3 1     企业     0.0291
#> 4 1     保护     0.0288
#> 5 1     合规     0.0215
```

查看每个文档的主题

- 找出每个主题的 TOP 文档

```
topic_texts = function(topic) {
  topic.docs = posterior(mdl)$topics[, topic]
  topic.docs = sort(topic.docs, decreasing = TRUE)
  rlt = head(topic.docs, 50)
  tibble(text = names(rlt), prob = rlt)
}
map_dfr(1:13, topic_texts, .id = "topic")
#> # A tibble: 182 x 3
#>   topic text          prob
#>   <chr> <chr>        <dbl>
#> 1 1     文档 6.txt   0.425
#> 2 1     文档 10.txt  0.0510
#> 3 1     文档 9.txt   0.0444
#> 4 1     文档 8.txt   0.0316
```

可视化

```
library(LDAvis)
library(slam)
dtm = dtm[slam::row_sums(dtm) > 0,]
phi = as.matrix(posterior(mdl)$terms)
theta = as.matrix(posterior(mdl)$topics)
vocab = colnames(phi)
doc.length = row_sums(dtm)
term.freq = col_sums(dtm)[match(vocab, colnames(dtm))]
json = createJSON(phi = phi, theta = theta, vocab = vocab,
                  doc.length = doc.length,
                  term.frequency = term.freq)
serVis(json)
```

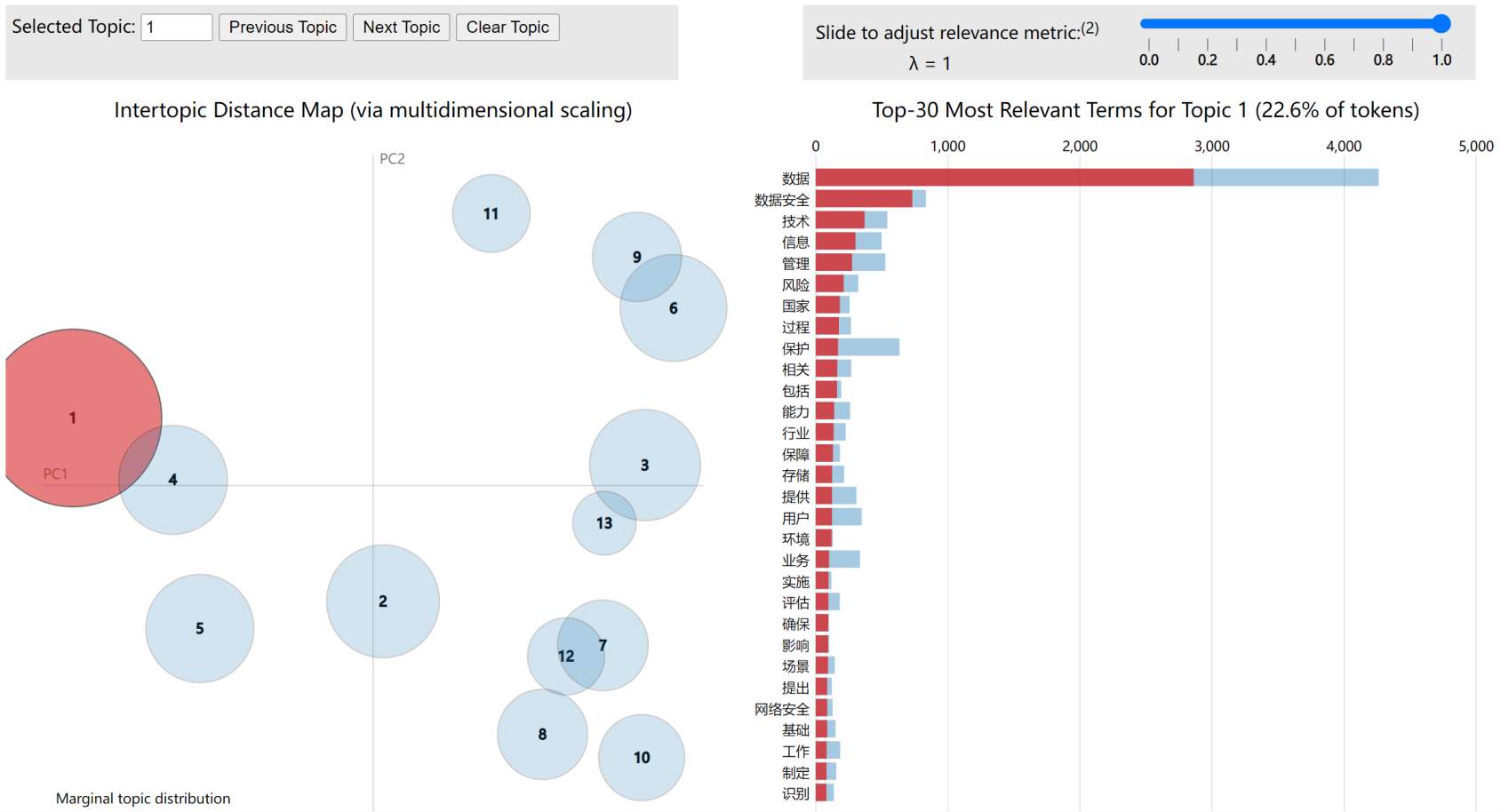

Selected Topic: 1
Previous Topic
Next Topic
Clear Topic
Slide to adjust relevance metric:(2)
λ = 1
0.0
0.2
0.4
0.6
0.8
1.0
Intertopic Distance Map (via multidimensional scaling)
PC2
PC1
1
2
3
4
5
6
7
8
9
10
11
12
13
Marginal topic distribution
Top-30 Most Relevant Terms for Topic 1 (22.6% of tokens)
0
1,000
2,000
3,000
4,000
5,000
数据
数据安全
技术
信息
管理
风险
国家
过程
保护
相关
包括
能力
行业
保障
存储
提供
用户
环境
业务
实施
评估
确保
影响
场景
提出
网络安全
基础
工作
制定
识别

图 4: LDA 模型可视化

- 对应关系

```
library(jsonlite)
df = fromJSON(json)
df$topic.order
```

五. 情感分析

就是读入情感词典，将文档分词与情感词典做内连接，再按文档或文档级变量分组汇总计算情感得分。

主要参考文献

[1] 张丹. 用结构化数据的方式来管理文本. 第 14 届中国 R 会, 2021.

[2] Kenneth Benoit, Stefan Müller. Text Analysis Using R. https://quanteda.github.io/Text-Analysis-Using-R/, 2022.

[3] Kohei Watanabe, Stefan Müller. QUANTEDA TUTORIALS. https://tutorials.quanteda.io/, 2022.

[4] Julia Silge, David Robinson. Text Mining with R: A Tidy Approach. https://www.tidytextmining.com/, 2022.