DISCOVERY

巫小珍

2022-08-11

目录

5 预测

5.1 文本数据

由于《联邦党人文集》是由多篇作者合著，且是笔名，所以我们可以通过爬取文本数据去分析每篇文章是谁写的。

使用的包：tm、SnowballC、wordcloud

步骤：

- 预处理：消除文字之间空白、数字、特定词、标点
- 抓取主干词
- 文本——项矩阵：计算项频率
- 生成词云图
- 计算逆文本频率度量：文本中各个项的重要程度的工具

  - 基于项频率的文本分析依赖于词包假设，忽略了词的顺序。为了度量文本中一个项的重要程度，我们可以计算项频率——逆文本频率，它代表以文本数反向加权后出现的项出现的相对频率。

处理代码：

- 加载程序包：`library(tm, SnowballC)`
- 加载语料和文本：`VCorpus(DirSource(directory, pattern))`
- 进行多种自然语言的处理：`tm_map(文本名字, function)`
  - `content_transformer(tolower)`：转换为小写。`tolower()` 并不是 tm 程序包中的函数，所以用包装函数 `content_transformer()`。
  - `removeNumbers()`：去掉数字。
  - `stripWhitespace()`：去掉空白。
  - `removeWords()`：去掉特定词。可以用 `stopwords()` 函数获取输入语言的停词表。
  - `removePunctuation()`：去掉标点
  - `stemDocument()`：根据某个词语生成主干词
- 列举选中文件的具体文本内容：`cotent(corpus[[number]])`
- 生成一维文本——项数据：`DocumentTermMatix()`
  - R 列出的是摘要而不是矩阵本身
  - Non-/ sparse entries：不是空白的数据量
  - Sparsity：输入项为零的比例
  - Maximal term length：矩阵项加权后的最大项的长度和数量
  - Weighting：tf 统计
- `inspect()`：获取具体的文本信息
- `stemCompletion()`：恢复主干词的原貌
- `weightTfIdf()`：计算逆文本频率 (tf-idf) 值，输入项为文本——项矩阵

5.1.1 《联邦党人文集》悬而未决的作者问题

```
## load two required libraries
library(tm, SnowballC)
```

```
## 载入需要的程辑包：NLP
```

```
## load the raw corpus
corpus.raw <- VCorpus(DirSource(directory = "Datasets/federalist",
                                pattern = "fp"))
corpus.raw
```

```
## <<VCorpus>>
## Metadata:  corpus specific: 0, document level (indexed): 0
## Content:  documents: 85
```

```
## make lower case
corpus.prep <- tm_map(corpus.raw, content_transformer(tolower))
## remove the white space
corpus.prep <- tm_map(corpus.raw, stripWhitespace)
## remove punctuaion
corpus.prep <- tm_map(corpus.raw, removePunctuation)
## remove numbers
corpus.prep <- tm_map(corpus.raw, removeNumbers)
## stopwords of English
head(stopwords("english"))
```

```
## [1] "i"      "me"     "my"     "myself" "we"     "our"
```

```
## remove stop words
corpus <- tm_map(corpus.prep, removeWords, stopwords("english"))
## finally stem remaining words
corpus <- tm_map(corpus, stemDocument)
```

```
## the output is truncated here to save space
content(corpus[[10]])  # essay no.10
```

5.1.2 文本——项矩阵

```
dtm <- DocumentTermMatrix(corpus)
dtm
```

```
## <<DocumentTermMatrix (documents: 85, terms: 10863)>>
## Non-/sparse entries: 55109/868246
## Sparsity           : 94%
```

```
## Maximal term length: 21
## Weighting          : term frequency (tf)
```

```
inspect(dtm[1:5, 1:8])
```

```
## <<DocumentTermMatrix (documents: 5, terms: 8)>>
## Non-/sparse entries: 1/39
## Sparsity           : 98%
## Maximal term length: 9
## Weighting          : term frequency (tf)
## Sample             :
##           Terms
## Docs       -day, -day. -grasp -industri -match -morrow -morrow. -scrupul
##   fp01.txt     0     0      0         0      0       0        0        1
##   fp02.txt     0     0      0         0      0       0        0        0
##   fp03.txt     0     0      0         0      0       0        0        0
##   fp04.txt     0     0      0         0      0       0        0        0
##   fp05.txt     0     0      0         0      0       0        0        0
```

```
## 转化成标准矩阵
dtm.mat <- as.matrix(dtm)
```

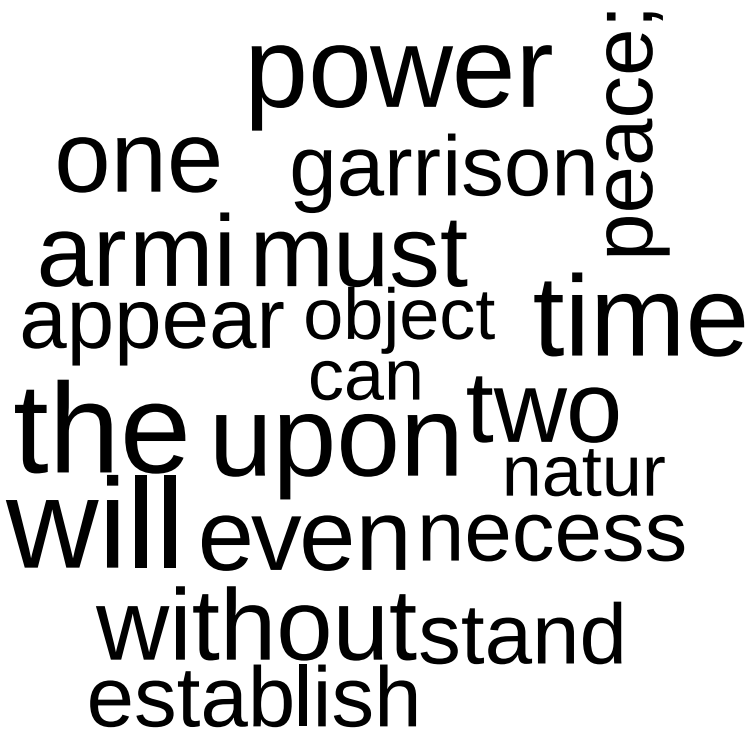
5.1.3 挖掘主题

```
library(wordcloud)
```

```
## 载入需要的程辑包: RColorBrewer
```

```
wordcloud(colnames(dtm.mat), dtm.mat[12, ], max.word = 20) # essay no .12
```


```
wordcloud(colnames(dtm.mat), dtm.mat[24, ], max.word = 20) # essay n0 .24
```




```
dtm.tfidf <- weightTfIdf(dtm) # tf-idf calculation
dtm.tfidf.mat <- as.matrix(dtm.tfidf) # convert to matrix
## 10 most important words for paper no. 12
head(sort(dtm.tfidf.mat[12, ], decreasing = TRUE), n = 10)
```

```
##     contraband         patrol          excis            tax            per
##     0.01815691     0.01815691     0.01532405     0.01428729     0.01366694
##       revenue.         trade.      commerce,         revenu communication,
##     0.01366694     0.01366694     0.01057349     0.01057349     0.01021604
```

```
head(sort(dtm.tfidf.mat[24, ], decreasing = TRUE), n = 10)
```

```
##   garrison     peace;  dock-yard     spain.       armi settlement    western
## 0.03170804 0.02395934 0.01502788 0.01502788 0.01371601 0.01268321 0.01266836
##   frontier     nearer       corp
## 0.01131167 0.01131167 0.01033855
```

```r
k <- 4  # number of clusters
## subset the Federalist papers written by Hamilton
hamilton <- c(1, 6:9, 11:13, 15:17, 21:36, 59:61, 65:85)
dtm.tfidf.hamilton <- dtm.tfidf.mat[hamilton, ]
## run k-means
km.out <- kmeans(dtm.tfidf.hamilton , centers = k)
km.out$iter
```

```
## [1] 3
```

```r
# check the convergence; number of iterations may vary
```

聚类的相关操作：输出聚类重心中 10 个最重要的项

```r
## label each centroid with corresponding terrm
colnames(km.out$centers) <- colnames(dtm.tfidf.hamilton)
for (i in 1:k) {
  cat("CLUSTER", i, "\n")
  cat("Top 10 words: \n")
  print(head(sort(km.out$centers[i, ], decreasing = TRUE), n = 10))
  cat("\n")
  cat("Federalist Paper classified:\n")
  print(rownames(dtm.tfidf.hamilton)[km.out$cluster == i])
  cat("\n")

}
```

```
## CLUSTER 1
## Top 10 words:
##       court        upon      presid    governor     senate,       senat
## 0.006160208 0.003855607 0.002586958 0.002215809 0.002206264 0.002187672
##      suprem       claus         tax        juri
## 0.002153387 0.002087159 0.002072636 0.002043073
##
## Federalist Paper classified:
##  [1] "fp01.txt" "fp07.txt" "fp09.txt" "fp15.txt" "fp16.txt" "fp17.txt"
##  [7] "fp21.txt" "fp22.txt" "fp23.txt" "fp27.txt" "fp30.txt" "fp31.txt"
## [13] "fp32.txt" "fp33.txt" "fp34.txt" "fp35.txt" "fp36.txt" "fp59.txt"
## [19] "fp60.txt" "fp61.txt" "fp65.txt" "fp66.txt" "fp68.txt" "fp69.txt"
```

```
## [25] "fp70.txt" "fp71.txt" "fp72.txt" "fp73.txt" "fp75.txt" "fp76.txt"
## [31] "fp77.txt" "fp78.txt" "fp79.txt" "fp80.txt" "fp81.txt" "fp82.txt"
## [37] "fp83.txt" "fp84.txt" "fp85.txt"
##
## CLUSTER 2
## Top 10 words:
##        armi    militari    militia,     militia   disciplin    commerci
## 0.008365183 0.007594486 0.005708095 0.005228487 0.004179071 0.003908762
##    garrison confederaci       stand    dominion
## 0.003658576 0.003326627 0.003190205 0.002991235
##
## Federalist Paper classified:
##  [1] "fp06.txt" "fp08.txt" "fp11.txt" "fp12.txt" "fp13.txt" "fp24.txt"
##  [7] "fp25.txt" "fp26.txt" "fp28.txt" "fp29.txt"
##
## CLUSTER 3
## Top 10 words:
##        vacanc        recess         claus          fill   established
##    0.05427482    0.03618321    0.02949959    0.02949959    0.02403522
##     otherwise       session appointments;      clauses,       appoint
##    0.02403522    0.02204695    0.02028522    0.02028522    0.01767127
##
## Federalist Paper classified:
## [1] "fp67.txt"
##
## CLUSTER 4
## Top 10 words:
##            pardon         clemency.             crime           treason
##        0.04641464        0.02698691        0.02031338        0.01856586
##             plead             sedit             merci "commander--chief
##        0.01721037        0.01610286        0.01516647        0.01349345
##          "repriev            benign
##        0.01349345        0.01349345
##
## Federalist Paper classified:
## [1] "fp74.txt"
```

5.1.4 作者预测

我们聚焦形容词、副词、介词和连词的使用去预测作者。我们需要使用没有被提取主干词的文本库 corpus.prep。首先计算项频率（每 1000 个词），然后用得到的项——频率矩阵生成只包括这些词的子集。

```
## document-term matrix converted to matrix for manipulation
dtm1 <- as.matrix(DocumentTermMatrix(corpus.prep))
tfm <- dtm1 / rowSums(dtm1) * 1000
## words of interest

words <- c("although", "always", "commonly", "consequently", "considerable",
           "enough", "there", "upon", "while", "whilst")
## select only these words
tfm <- tfm[, words]
## essays written by Mardison
madison <- c(10, 14, 37:48, 58)

## average among Hadison/Madison essays
tfm.ave <- rbind(colSums(tfm[hamilton,])/ length(hamilton),
                 colSums(tfm[madison,])/ length(madison))
tfm.ave
```

```
##        although    always  commonly consequently considerable   enough    there
## [1,] 0.01741372 0.7488197 0.2617445   0.01123953    0.5275750 0.355302 4.355111
## [2,] 0.22886745 0.1996487 0.0000000   0.33861003    0.1594164 0.000000 1.019523
##           upon     while      whilst
## [1,] 4.3095429 0.3680435 0.007002801
## [2,] 0.1989489 0.0000000 0.377907805
```

基于”although”、“upon”、“consequently”、“whilst” 四个词的的频率作为线性回归模型的预测因子，其中因变量是文章的作者。我们首先讲线性回归模型拟合已知作者的文章来估计预估系数。将因变量编码为作者是汉密尔顿时等于，作者是麦迪逊时等于-1，接着构建一个数据框对象。

```
author <- rep(NA, nrow(dtm1))
author[hamilton] <- 1
author[madison] <- -1
## data frame for regression
author.data <- data.frame(author = author[c(hamilton, madison)],
                          tfm[c(hamilton, madison), ])
hm.fit <- lm(author ~ upon + there + consequently + whilst,
```

```
              data = author.data)
hm.fit
```

```
##
## Call:
## lm(formula = author ~ upon + there + consequently + whilst, data = author.data)
##
## Coefficients:
##  (Intercept)          upon         there  consequently        whilst
##      -0.29916       0.17368       0.09875      -0.29774      -0.78111
```

```
hm.fitted <- fitted(hm.fit)
sd(hm.fitted)
```

```
## [1] 0.7142683
```

5.1.5 交叉验证

留一交叉验证是样本外预测的程序。将一个观测值去掉并用模型拟合剩下的数据，然后预测它的结果变量值。对样本的每一个观测值重复这一过程，并计算分类误差。

交叉验证不依赖于样本内预测来评估模型的准确性。

在 R 中我们使用循环语句进行交叉验证。

```
## 样本内预测的基础上的分类成功率
## proportion of correctly classified essays by Hamilton
mean(hm.fitted[author.data$author == 1] > 0)
```

```
## [1] 0.9803922
```

```
## proportion of correctly classified essays by Madison
mean(hm.fitted[author.data$author == -1] < 0)
```

```
## [1] 1
```

```
## 交叉验证---基于样本外预测
n <- nrow(author.data)
hm.classify <- rep(NA, n)
for(i in 1:n){
```

```
  ## fit the model to the data after removing the ith observation
  sub.fit <- lm(author ~ upon + there + consequently + whilst,
                data = author.data[-i, ])
  ## predict the authorship for the ith observation
  hm.classify[i] <- predict(sub.fit, newdata = author.data[i, ])

}
mean( hm.classify[author.data$author == 1] > 0)
```

```
## [1] 0.9803922
```

```
mean( hm.classify[author.data$author == -1] < 0)
```

```
## [1] 1
```

```
## 拟合未知作者的 11 篇文章
disputed <- c(49, 50:57, 62, 63)
tf.disputed <- as.data.frame(tfm[disputed, ])
pred <- predict(hm.fit, newdata = tf.disputed)
pred
```

```
##    fp49.txt    fp50.txt    fp51.txt    fp52.txt    fp53.txt    fp54.txt
## -1.00764340 -0.09673060 -1.52241505 -0.29915634 -0.64904444 -0.20018028
##    fp55.txt    fp56.txt    fp57.txt    fp62.txt    fp63.txt
##  0.01754377 -0.69658770 -1.46303583 -0.63276367 -0.29534162
```

5.2 网络数据

使用到的包：igraph

`as.matrix()`：转换成矩阵

`rowSums()`：计算按列计算的总数

`colSums()`：计算按行计算的总数

`graph.adjacency()`：绘制网络，输入为邻接矩阵

- mode：undirected directed
- diag：FALSE TRUE

`plot()`：画为网络图，输入为 igraph 对象。

- vertex.size：调整每个节点的大小
- vertex.color：调整每个点的颜色
- vertex.label：具体说明每个节点的标签
- edge.arrow.size：调整每个点箭头的大小
- edge.width：调整每条边的宽度
- 更多细节可以参见 igraph.plotting

`degree()`：量化在多大程度上每个节点和其他节点相连并出现在图中心。计算边的数量或度值

- mode：in（入度值）out（出度值）total（默认值）

`closeness()`：接近值。图中一个节点距离其他节点有多远。

- mode：in（入度值）out（出度值）total（默认值）

`betweenness()`：中介值。一个节点如果负责联结其他所有点，则被认为处在中心。

- directed：FALSE TRUE

`order()`：排序，返回一个索引向量

- decreasing：TRUE（按从大到小）FALSE（按从小到大）

`sort()`：排序，返回一个排序向量

- decreasing：TRUE（按从大到小）FALSE（按从小到大）

`page.rank()`: 计算网页排名值。命名为 vector 的向量是网页排名值。

- directed：FALSE TRUE

5.2.1 文艺复兴时期佛罗伦萨的婚姻网络

```
florence <- read.csv("Datasets/florentine.csv", row.names = "FAMILY")
florence <- as.matrix(florence)

florence[1:5, 1:5]
```

```
##           ACCIAIUOL ALBIZZI BARBADORI BISCHERI CASTELLAN
## ACCIAIUOL         0       0         0        0         0
## ALBIZZI           0       0         0        0         0
## BARBADORI         0       0         0        0         1
## BISCHERI          0       0         0        0         0
## CASTELLAN         0       0         1        0         0
```

```
rowSums(florence)
```

```
## ACCIAIUOL   ALBIZZI BARBADORI  BISCHERI CASTELLAN    GINORI  GUADAGNI LAMBERTES
##         1         3         2         3         3         1         4         1
##    MEDICI     PAZZI   PERUZZI     PUCCI   RIDOLFI  SALVIATI   STROZZI TORNABUON
##         6         1         3         0         3         2         4         3
```

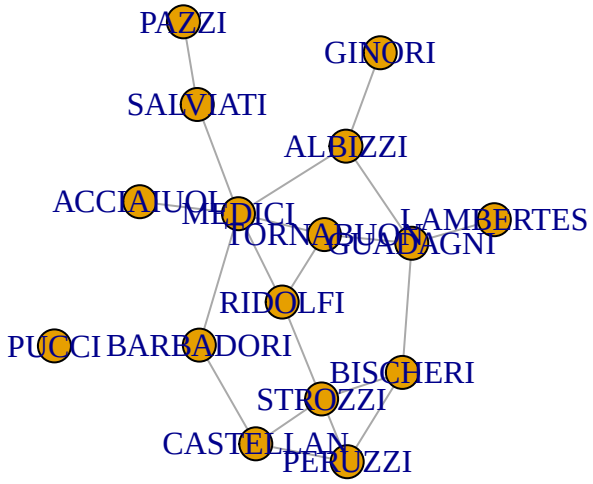
5.2.2 无方向的绘图和中心性度量

中心性度量包括度值、接近值和中介值。这些度量评估了多大程度上每个节点在一个图中扮演了中心角色。

```
library("igraph")
```

```
##
## 载入程辑包: 'igraph'

## The following objects are masked from 'package:stats':
##
##     decompose, spectrum

## The following object is masked from 'package:base':
##
##     union
```

```
florence <- graph.adjacency(florence, mode = "undirected", diag = FALSE)
plot(florence)
```




```
degree(florence)
```

```
## ACCIAIUOL   ALBIZZI BARBADORI  BISCHERI CASTELLAN    GINORI  GUADAGNI LAMBERTES
##         1         3         2         3         3         1         4         1
##    MEDICI     PAZZI   PERUZZI     PUCCI   RIDOLFI  SALVIATI   STROZZI TORNABUON
##         6         1         3         0         3         2         4         3
```

```
closeness(florence)
```

```
##  ACCIAIUOL    ALBIZZI  BARBADORI   BISCHERI  CASTELLAN     GINORI   GUADAGNI
## 0.02631579 0.03448276 0.03125000 0.02857143 0.02777778 0.02380952 0.03333333
##  LAMBERTES     MEDICI      PAZZI    PERUZZI      PUCCI    RIDOLFI   SALVIATI
## 0.02325581 0.04000000 0.02040816 0.02631579        NaN 0.03571429 0.02777778
##    STROZZI  TORNABUON
## 0.03125000 0.03448276
```

```r
1 / (closeness(florence) * 15) ## 平均边数
```

```
## ACCIAIUOL   ALBIZZI BARBADORI  BISCHERI CASTELLAN    GINORI  GUADAGNI LAMBERTES
##  2.533333  1.933333  2.133333  2.333333  2.400000  2.800000  2.000000  2.866667
##    MEDICI     PAZZI   PERUZZI     PUCCI   RIDOLFI  SALVIATI   STROZZI TORNABUON
##  1.666667  3.266667  2.533333       NaN  1.866667  2.400000  2.133333  1.933333
```

```r
betweenness(florence)
```

```
## ACCIAIUOL   ALBIZZI BARBADORI  BISCHERI CASTELLAN    GINORI  GUADAGNI LAMBERTES
##  0.000000 19.333333  8.500000  9.500000  5.000000  0.000000 23.166667  0.000000
##    MEDICI     PAZZI   PERUZZI     PUCCI   RIDOLFI  SALVIATI   STROZZI TORNABUON
## 47.500000  0.000000  2.000000  0.000000 10.333333 13.000000  9.333333  8.333333
```

```r
# plot(florence, vertex.size = closeness(florence), main = "Closeness")
plot(florence, vertex.size = betweenness(florence), main = "Betweenness")
```

5.2.3 推特关注网络

```
twitter <- read.csv("Datasets/twitter-following.csv",
                    stringsAsFactors = FALSE)
## stringsAsFactors = FALSE 不将其作为因子进行获取
senator <- read.csv("Datasets/twitter-senator.csv",
                    stringsAsFactors = FALSE)
n <- nrow(senator)
twitter.mat <- matrix(0, nrow = n, ncol = n)
colnames(twitter.mat) <- rownames(twitter.mat) <- senator$screen_name
for(i in 1:nrow(twitter)){
  twitter.mat[twitter$following[i], twitter$followed[i]] <-  1

}
twitter.adj <- graph.adjacency(twitter.mat, mode = "directed",
                               diag = FALSE)
```

5.2.4 有方向的图和中心性

有方向的网络中的有三种中心性进行定义。度值的两种方法为：一个节点收到的边数总和称为入度值；一个节点发出的边数总和称为出度值。

入射路径的入度值和接近值是一致的，出射路径的出度值和接近值是一致的。

```
senator$indegree <- degree(twitter.adj, mode = "in")
senator$outdegree <- degree(twitter.adj, mode = "out")
in.order <- order(senator$indegree, decreasing = TRUE)
out.order <- order(senator$outdegree, decreasing = TRUE)
senator[in.order[1:3],]
```

```
##       screen_name             name party state indegree outdegree
## 51  SenJohnMcCain      John McCain     R    AZ       64        15
## 57  lisamurkowski   Lisa Murkowski     R    AK       60        87
## 18 SenatorCollins Susan M. Collins     R    ME       58        79
```

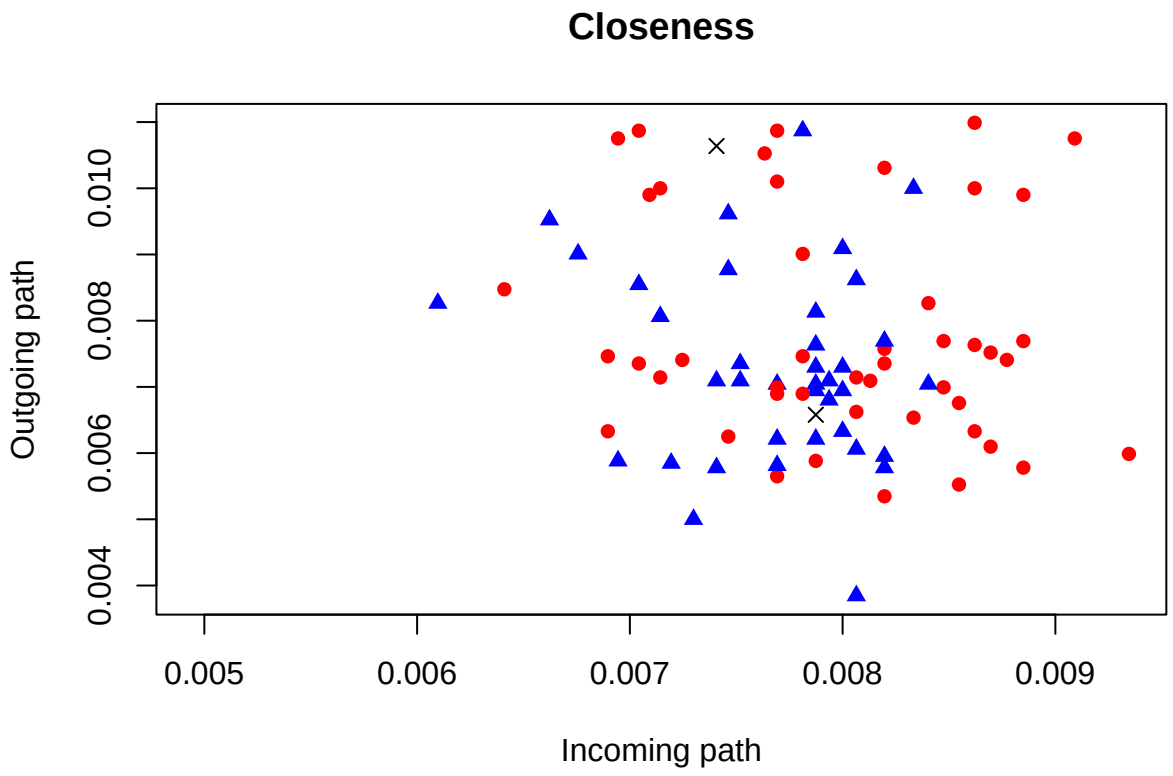
```
senator[out.order[1:3],]
```

```
##       screen_name                 name party state indegree outdegree
## 37  SenDeanHeller          Dean Heller     R    NV       55        89
```

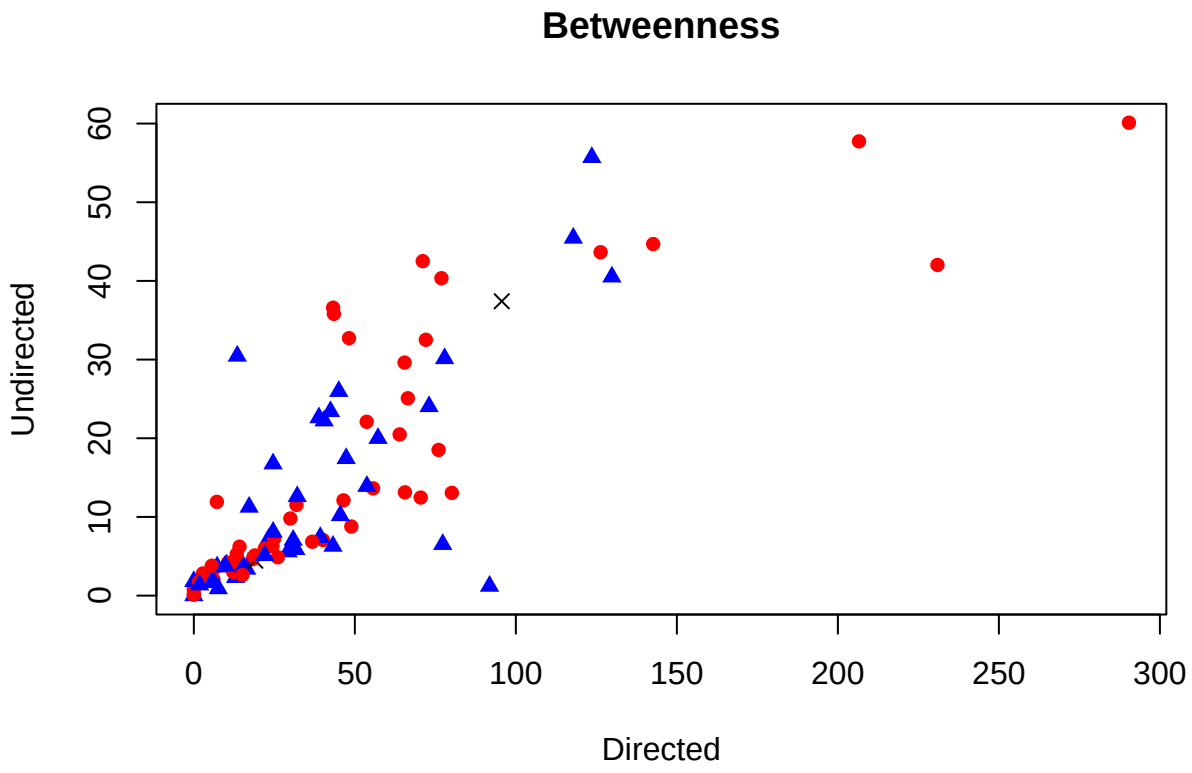
```
## 21    SenBobCasey Robert P. Casey, Jr.     D    PA       43        88
## 65 sendavidperdue          David Perdue     R    GA       30        88
```

```
## 蓝色三角形表示民主党人，红色圆形表示共和党人，黑色十字为独立人士
n <- nrow(senator)
col <- rep("red", n)
col[senator$party == "D"] <- "blue"
col[senator$party == "I"] <- "black"
pch <- rep(16, n)
pch[senator$party == "D"] <- 17
pch[senator$party == "I"] <- 4
```

```
## plot for comparing two closeness measures
plot(closeness(twitter.adj, mode = "in"),
     closeness(twitter.adj, mode = "out"), pch = pch , col = col,
     main = "Closeness", xlab = "Incoming path", ylab = "Outgoing path")
```



```
plot(betweenness(twitter.adj, directed =  TRUE),
     betweenness(twitter.adj, directed =  FALSE), pch = pch, col = col,
     main = "Betweenness", xlab = "Directed", ylab = "Undirected")
```

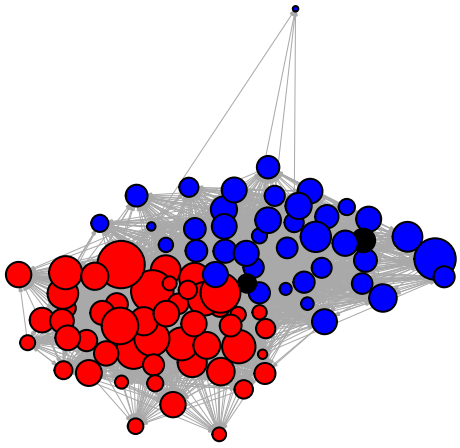


网络排名是度量中心性的另一种方式。网页排名是一种将每一个节点” 投票” 平均分配给与其联结的点，是度量中心度的迭代算法。网站排名数之和为 1。用 `page.rank()` 函数计算。

$$j = (1-d)/n + d * \sum(Aij * \quad i/\ i)$$

d 为一个指定的常数，经常设为 0.85。n 为节点的数量。

```
senator$pagerank <- page.rank(twitter.adj)$vector
plot(twitter.adj, vertex.size = senator$pagerank * 1000,
     vertex.color = col, vertex.label = NA, edge.arrow.size = 0.1,
     edge.width = 0.5)
```



```
while(condition){

LOOP CONTENTS HERE

}
```

```
## 定义一个在每次迭代中更新网页排名值的函数
Pagerank <-  function(n, A, d, pr){
  deg <- degree(A, mode = "out")
  for(j in 1:n){
    pr[j] <-(1-d) / n +d * sum(A[, j] * pr / deg)
  }
  return(pr)
}
nodes <- 4
adj <- matrix(c(0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0 ),
              ncol = nodes, nrow = nodes, byrow = TRUE)
adj
```

```
##      [,1] [,2] [,3] [,4]
```

```
## [1,]    0    1    0    1
## [2,]    1    0    1    0
## [3,]    0    1    0    0
## [4,]    0    1    0    0
```

```
adj <- graph.adjacency(adj)

d <- 0.85
pr <- rep(1/nodes, nodes)
## while 循环来迭代执行算法直到满足收敛标准。
## 对于收敛标准，我们使用 0.001 为连续的两次交互中网页排名值之差的最大绝对值
## 我们对节点之间使用同样的网页排名值作为初始值
diff <- 100
while(diff < 0.001){
  pr.pre <- pr
  pr <- Pagerank(n = nodes, A = adj, d = d, pr = pr)
  diff <- max(abs(pr - pr.pre))
}
```

5.3 空间数据

5.3.1 1854 年伦敦爆发霍乱

5.3.2 R 中的空间数据

使用的包：maps

data()：加载数据

map()：加载地图

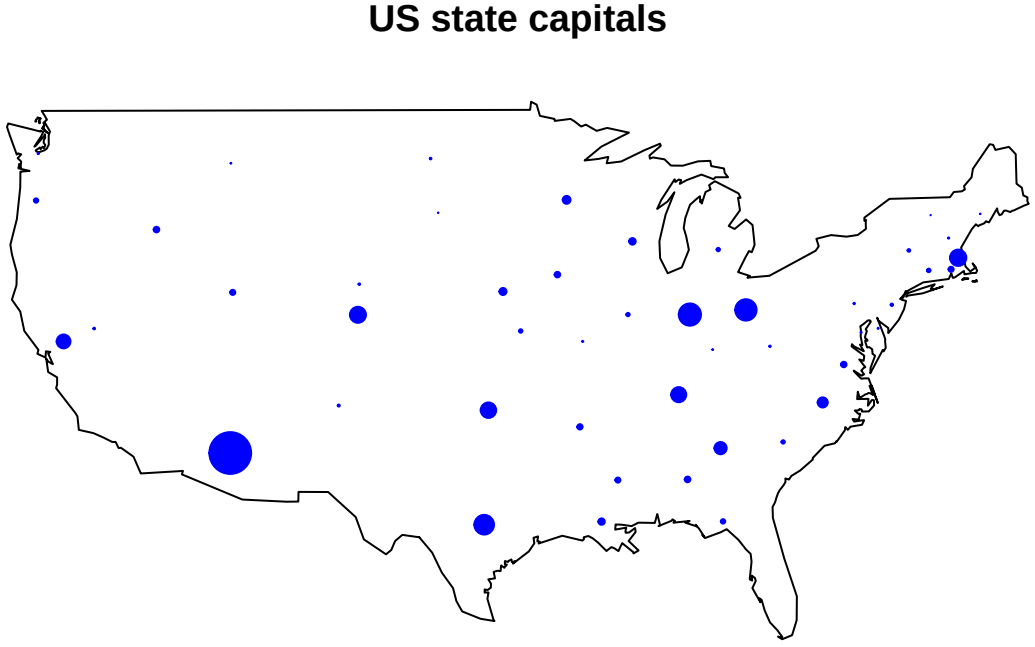
- database：数据库
- region：地区

空间数据包括空间模式的信息，并可以通过地图可视化。地图上的空间点数据代表了事件，**空间多边形数据**代表了由点联结的地理区域。

```
library(maps)
data(us.cities)
head(us.cities)
```

```
##          name country.etc    pop   lat    long capital
## 1 Abilene TX          TX 113888 32.45  -99.74       0
## 2   Akron OH          OH 206634 41.08  -81.52       0
## 3 Alameda CA          CA  70069 37.77 -122.26       0
## 4  Albany GA          GA  75510 31.58  -84.18       0
## 5  Albany NY          NY  93576 42.67  -73.80       2
## 6  Albany OR          OR  45535 44.62 -123.09       0
```

```
map(database = "usa")
capitals <- subset(us.cities, capital == 2) ## subset state capitals
## add points porportional to population using latitude and longitude
points(x = capitals$long, y = capitals$lat, col = "blue",
       cex = capitals$pop / 500000, pch = 19)
title("US state capitals") ## add a title
```




```
map(database = "state", regions = "California")
cal.cities <- subset(us.cities, subset = (country.etc == "CA"))
sind <- order(cal.cities$pop, decreasing = TRUE)
```

```
top7 <- sind[1:7]
points(x = cal.cities$long[top7], y = cal.cities$lat[top7], pch = 19)
text(x = cal.cities$long[top7] + 2.25, y = cal.cities$lat[top7],
     label = cal.cities$name[top7])
title("Largest cities of California")
```

5.3.3 R 中的色彩

`colors()`：获取 657 种颜色。

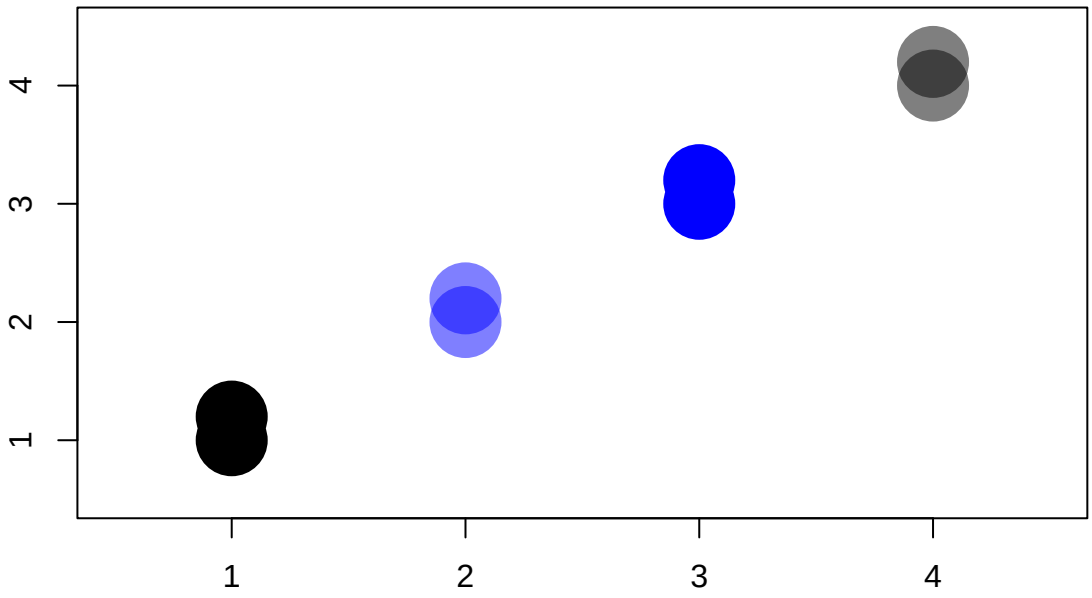
```
rgb(red = 1, green = 0, blue = 0)
```

```
rgb(red = c(0.5, 1), green = c(0, 1), blue = c(0.5, 0))
```

```
rgb(red = 1, green = 0, blue = 0, alphan = 0.5)
```

```
plot(x = c(1, 1), y = c(1, 1.2), xlim = c(0.5, 4.5), ylim = c(0.5, 4.5),
     pch = 16, cex = 5, ann = FALSE, col = "black")
points(x = c(3, 3), y = c(3, 3.2), pch = 16, cex = 5, col = "blue")
points(x = c(2, 2), y = c(2, 2.2), pch = 16, cex = 5,
```

```
       col = rgb(red = 0, blue = 1, green = 0, alpha = 0.5))
points(x = c(4,4), y = c(4, 4.2), pch = 16, cex = 5,
       col = rgb(red = 0, blue = 0, green = 0, alpha = 0.5))
```



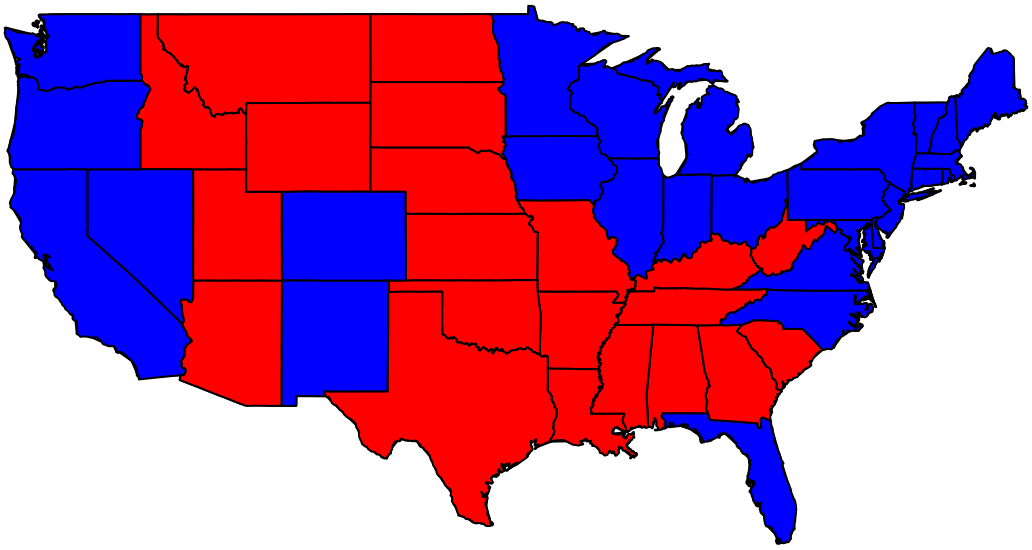
5.3.4 美国总统选举

计算两党获得选票的比例并配色。

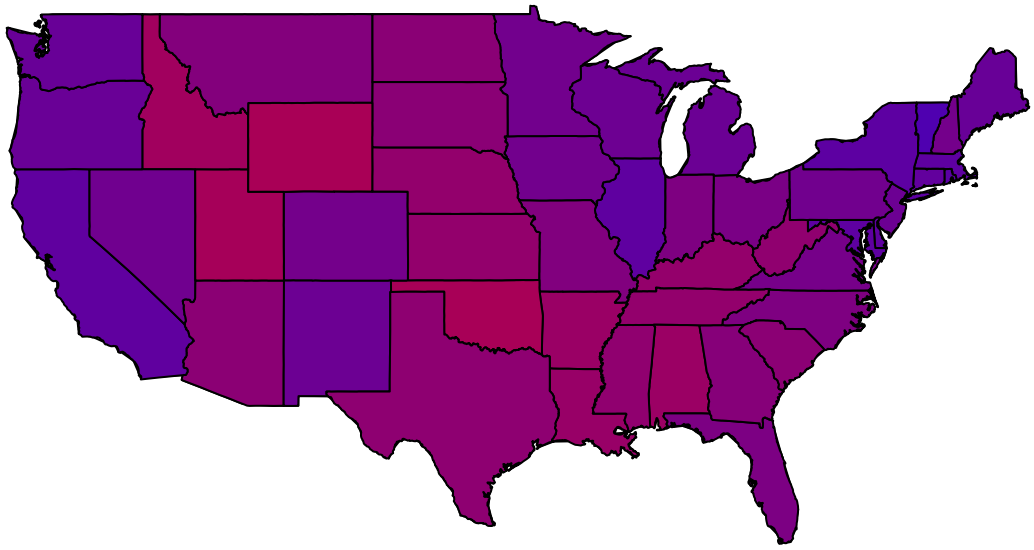
```
pres08 <- read.csv("Datasets/pres08.csv")
pres08$Dem <- pres08$Obama / (pres08$Obama + pres08$McCain)
pres08$Rep <- pres08$McCain / (pres08$Obama + pres08$McCain)
## color for California
cal.color <- rgb(red = pres08$Rep[pres08$state == "CA"],
                 blue = pres08$Dem[pres08$state == "CA"],
                 green = 0)
## California as a blue state
## fill 是为每一个州填充具体的颜色
map(database = "state", regions = "California", col = "blue", fill = TRUE)
```

```
## California as a purple state
map(database = "state", regions = "California", col = cal.color, fill = TRUE)
```

```
## USA as red and blue states
map(database = "state")
for(i in 1:nrow(pres08)){
  if ((pres08$state[i] != "HI") & (pres08$state[i] != "AK") &
      (pres08$state[i] != "DC")){
    map(database = "state", regions = pres08$state.name[i],
        col = ifelse(pres08$Rep[i] > pres08$Dem[i], "red", "blue"),
        fill = TRUE, add = TRUE)
  }

}
```



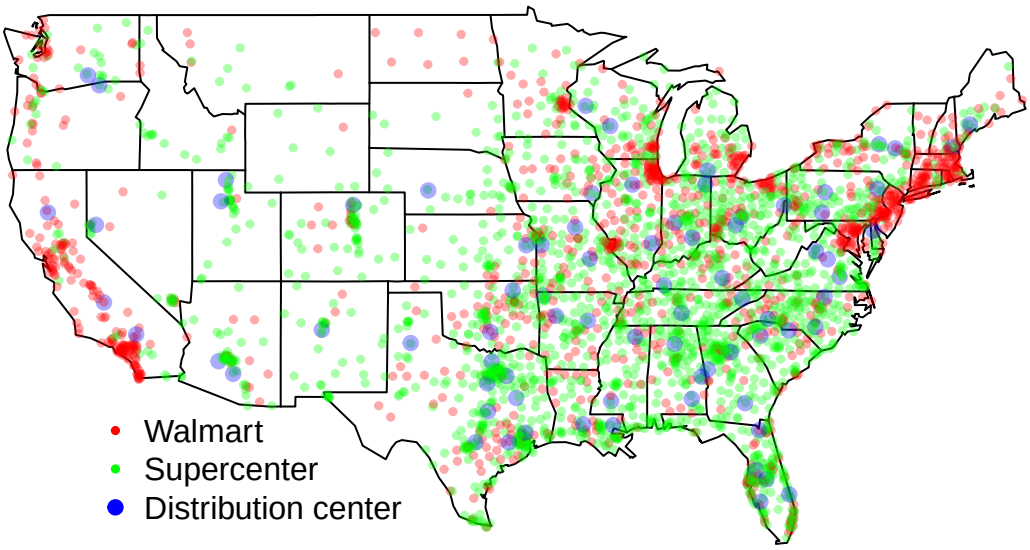
```
## USA as purple states
map(database = "state")
for(i in 1:nrow(pres08)){
  if ((pres08$state[i] != "HI") & (pres08$state[i] != "AK") &
      (pres08$state[i] != "DC")){
    map(database = "state", regions = pres08$state.name[i],
        col = rgb(red = pres08$Rep[i], blue = pres08$Dem[i], green = 0),
        fill = TRUE, add = TRUE)
  }

}
```



5.3.5 沃尔玛恶扩张

```
walmart <- read.csv("Datasets/walmart.csv")
walmart$storecolors <- NA
walmart$storecolors[walmart$type == "Wal-MartStore"] <-
  rgb(red = 1, green = 0, blue = 0, alpha = 1/3 )
walmart$storecolors[walmart$type == "SuperCenter"] <-
  rgb(red = 0, green = 1, blue = 0, alpha = 1/3 )
walmart$storecolors[walmart$type == "DistributionCenter"] <-
  rgb(red = 0, green = 0, blue = 1, alpha = 1/3 )
walmart$storesizes <- ifelse(walmart$type == "DistributionCenter", 1, 0.5)
## map with legend
map(database = "state")
points(walmart$long, walmart$lat, col = walmart$storecolors,
       pch = 19, cex = walmart$storesizes)
legend(x = -120, y = 32, bty = "n",
       legend = c("Walmart","Supercenter", "Distribution center"),
```

```
       col = c("red", "green", "blue"), pch = 19,
       pt.cex = c(0.5, 0.5, 1))
```



5.3.6 R 中的动画

使用的包：animation

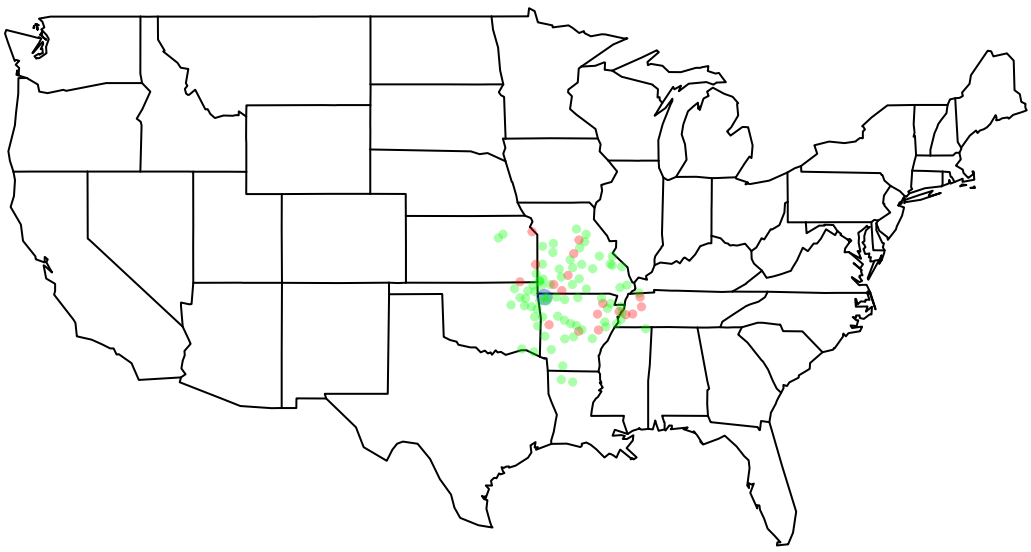
`saveHTML()`：网络浏览器播放这一动画。

- htmlfile：网页文件的名称
- outdir：设置存储生成文件的目录名称
- title：用来设置动画的名称
- autobrowse：设置输出结果是否在浏览器上播放

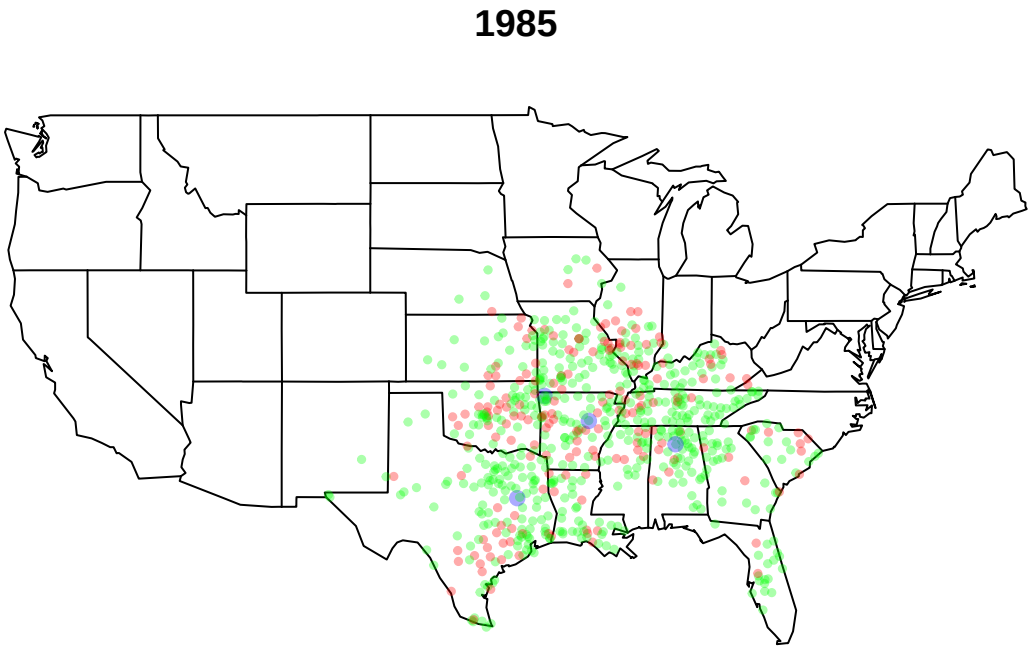
其他用于存储的文件函数还有 `saveLatex()`、`saveVideo()` 函数

根据给定日期生成地图。可以构造函数，绘制地图。

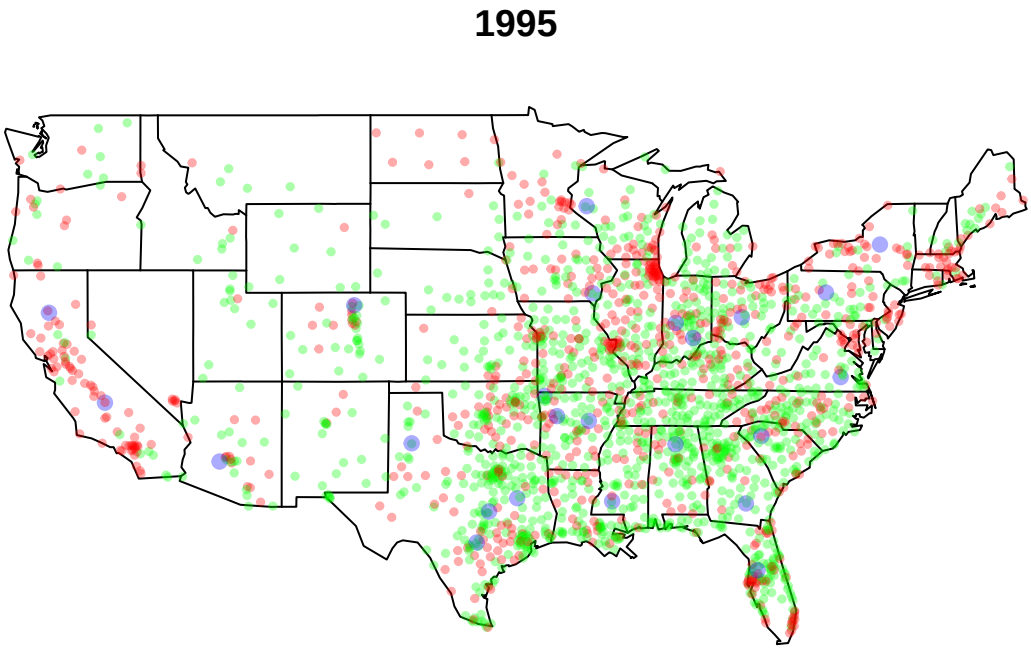
```
walmart.map <- function(data, date){
  walmart <- subset(data, subset = (opendate <= date))
  map(database = "state")
  points(walmart$long, walmart$lat, col = walmart$storecolors,
         pch = 19, cex = walmart$storesizes)
}

walmart$opendate <- as.Date(walmart$opendate)
walmart.map(walmart, as.Date("1974-12-31"))
title("1975")
```



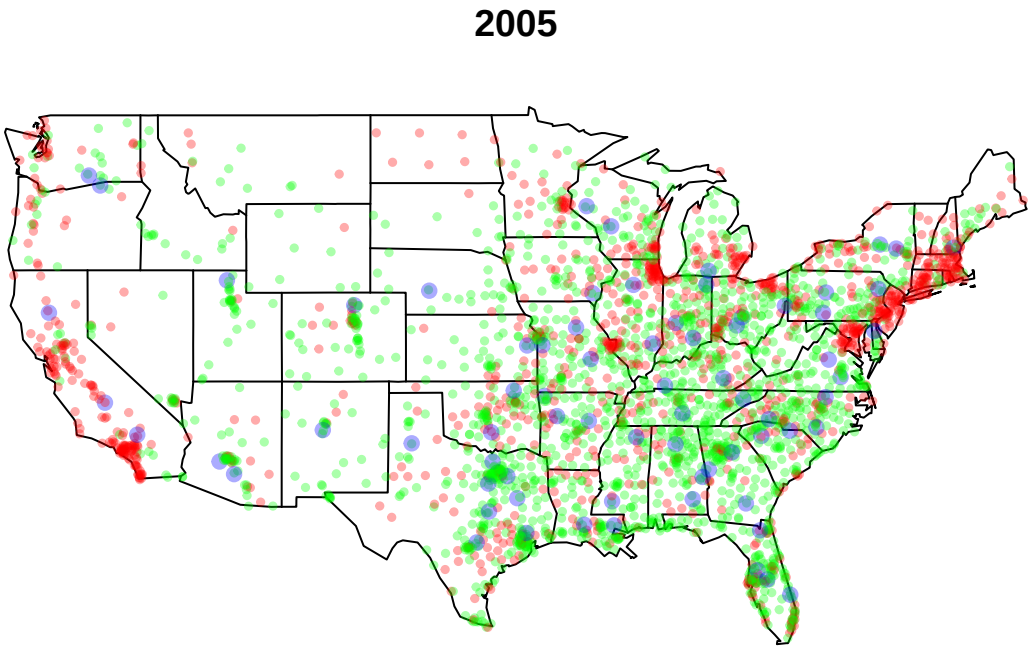

```
walmart.map(walmart, as.Date("1984-12-31"))
title("1985")
```




```
walmart.map(walmart, as.Date("1994-12-31"))
title("1995")
```




```
walmart.map(walmart, as.Date("2004-12-31"))
title("2005")
```



设置动画

```
n <- 25
dates <- seq(from = min(walmart$opendate), to = max(walmart$opendate),
             length.out = n)
library("animation")
saveHTML({
  for(i in 1:length(dates)){
    walmart.map(walmart, dates[i])
    title(dates[i])
  }

}, title = "Expansion of Walmart", htmlfile = "walmart.html", outdir = getwd(),
autobrowse = FALSE)
```

```
## HTML file created at: walmart.html
```

5.4 总结

5.5 习题

exercise1 ： 分析宪法的序言（见 exercise1.R)

- **question1：**理解宪法有什么不同。用两种方法去观察宪法语言。文本预处理——> 两种方法：文本——项和 tf-idf 加权的频数（注意要将初始值设置为默认的 FALSE）——> 生成数据矩阵——> 词云图

  参考：https://www.csdn.net/tags/NtTaAg4sMDMzMDEtYmxvZwO0O0OO0O0O.html

- **question2：**k-means 均值算法运用于每一行中，聚类数设置为 5 并可可视化结果，将其除于一个常数为 |a|——每一个行内的长度的平均值。

- **question3：**利用 question1 生成的文本——项的频数矩阵计算文本余弦相似度，并获取最相似的五个宪法

- **question4：**随着时间变化美国宪法对其他国家宪法的影响。按照十年为一类，分 1960-2010 年，分别计算每十年美国宪法与其他国家宪法相似度的平均值，并画图。

- **question5：**构建一个余弦相似度、有方向和权重的网络。生产一个比邻矩阵，i 和 j 为各个宪法，(i, j) 为第 i 个宪法与第 j 个宪法的余弦相似度。且如果第 i 个宪法是在与第 j 个宪法之前指定的的，则输入项为 0。并用 plot() 画图。

exercise2：国际贸易网络（见 exercise2.R)

- **question1：**有向无权重的网络进行分析。并计算网络密度。

- **question2：**分别计算 1900、1955 和 2009 年的基于度值、中介值和接近值的中心度。列出 top5 的国家。

- **question3：**有权重有向的网络分析。用 `graph.strength()` 函数计算权重，继续利用 `closeness()` 和 `betweenness()` 极端中介值和接近值。

- **question4：**对每一年的权重贸易网络单独应用 page-rank 算法。获取 top5（排序的方法获取）。并去寻找美国、英国、俄罗斯、日本和中国的 page-rank 的时间变化。

exercise3：制作美国总统选举跨时间的地图（见 exercise3.R)

- **question1：**可视化 2008 年美国大选中**马萨诸塞州和亚利桑那州**的两党投票比例，颜色按照两党投票比例中设置纯蓝色和纯红色比例变化。

- **question2：用 for 循环**可视化 2008 年美国大选中**每个郡**的两党投票比例

- **question3：动画可视化** 1960 年-2012 年美国大选中每个郡的两党投票比例。
- **question4：居民分割程度：**通过每个郡计算每个州的民主党和共和党的相异系数并在地图上显示。
- **question5：政治分割程度：**基于民主党在每个郡的选票比例，通过 `ineq（type = "Gini"）`计算每个州基尼系数并可视化。
- **question6：动画可视化** 1960 年-2012 年美国大选中**每个州**的政治分割程度。