R语言在CGED-Q JSL中的运用

陈俊
香港科技大学社会科学部
2025年5月(V2.1.1)

Six

数据的整理与匹配

01

筛选列

6.1 筛选列
6.1.1 `select()`函数

- 从数据库中筛选列的函数是`select()`，可以挑选出指定条件的列变量

- `select()`函数是`base R`（一个函数聚合项目）的一个筛选函数，其实在`tidyverse`项目中也有相关的筛选函数，但`base R`的筛选函数更加简短、更加快捷、便利理解，适合初学者使用，因此，我们这里使用的是`base R`的初阶筛选函数。`select()`的格式为：

```
`select(.data,...)`
```

  - `data`为数据集，…为需要选择的列，可以套用参数，该函数常用的参数有：
    `last_col()`:表示选择最后一列。
  - `starts_with()`:表示以什么开头的列
  - `ends_with()`:表示以什么结尾的列，
  - `contains()`:表示某列是否包含什么内容

6.1.2 保留核心变量

- 缙绅录数据库中有50余个变量，但有时候为了分析方便，我们往往只需要其中一部分变量，比如姓名、官职、籍贯、出身、任职地等重要变量，这时就需要用到`select()`函数。示例如下：

```
JSL1900_1912_clean %>%
    select(阳历年份,地区,机构一,官职一,姓,名,籍贯省,籍贯县,旗分,出身一) ->
 JSL1900_1912_simplify
```

代码说明

- `%>%`为管道符，可以将某一个数据集的行或列转移到另一个数据集，在第四章第三节介绍导出表格时用到过
- 利用管道符和`select()`函数，可以截取某一个数据集的部分变量生成新的数据集
- “阳历年份…出身一”为需要转移的列变量的名称，注意，这里不需要再指定数据集，因为管道符的有一个作用类似于`attach()`函数，运行`attach()`函数后，在之后的分析中就无需每次都指定代码集，RStudio会默认使用`attach()`函数指定的代码集，但很多使用者在数据集分析之后会忘记取消`attach()`函数中指定的代码集，在其他分析中还是套用原数据集影响分析结果，故不推荐使用
- `JSL1900_1912_simplify`为转移的/生成的数据集的名称

```
JSL1900_1912_clean %>%
    select(阳历年份,地区,机构一,官职一,姓,名,籍贯省,籍贯县,旗分,出身一) ->
 JSL1900_1912_simplify
```

02
筛选行

6.2 筛选行

6.2.1 `filter()`函数

- 筛选行的`filter()`函数，可以以挑选出指定条件的列，其用法与`select()`函数一样，格式为：

```
`filter(.data,...)`
```

6.2.2 筛选满足条件的行

```
JSL1900_1912_clean %>%
    filter((阳历年份numeric >= 1900)&
        (阳历年份numeric <= 1906)&
        (机构一 == "翰林院衙門")) -> JSL1900_1906_hanlinyuan
```

- “阳历年份numeric…”是筛选的条件，需要使用者注意连接符号的运用
- 无论是筛选行还是筛选列，都需要注意`%>%`管道符的使用，可以利用管道符将一个数据集中符合条件的列转移到另一个数据集
- 到这里，我们已经学习了两种按条件生成数据集的方法，一种是`subset()`函数，一种是dplyr包下的管道符与`filter()`函数连用，可以根据自己的需求和对函数的理解来选择适合自己的方法

03

匹配以整理数据集

6.3 匹配以整理变量
6.3.1 `merge()`函数

- 在第四章第二节我们运用到`factor()`函数来整理变量，使其更加简洁
- 但这种方法只适用于“籍贯省”这种记录类型比较少的变量，而“出身一”这一变量的记录类型会达到数百个，利用`factor()`一一输入对应无疑加大了工作量
- 面对这种记录类型多的变量，我们在整理时需要用到匹配函数。在`dplyr`包中，匹配函数是`merge()`，格式为：

```
`merge(x, y, ...)`
```

6.3.2 `merge()`函数的参数

- `x`为主数据集，`y`为副数据集，`…`为参数。常用的参数有：
- `by = …`表示选择主数据集和副数据集连接的列，当遇到主数据集和副数据集标题名称不同时，可以将by参数拆分为by.x和by.y参数，令by.x和by.y分别等于名称不同但内容有交集的公共列
- `all = …`表示全连接，可选TRUE或FALSE。当只需要保存主数据集的所有列而不需要副数据集的列时，令all.x = TRUE并且all.y = False，即为左连接；当只需要保存副数据集的所有列而不需要主数据集的列时，令all.x = False并且all.y = TRUE，即为右连接；当只需要主数据集和副数据集的交集，即令all = False，即为求交集，这也是R的默认选项
- `sort`表示`by`指定的列（即公共列）是否要排序
- `suffixes`表示指定除by外相同列名的后缀
- `incomparables`表示指定by中哪些单元不进行合并

6.3.3 整理“出身一”变量

- 现在，我们按照`merge()`函数的格式尝试整理“出身一”变量
- 导入匹配所需要的副数据集，这里我们用到的是整理“出身一”、“籍贯省”的副数据集
- 该数据集在第一章“缙绅录的下载”中讲到，请使用者在香港科技大学数据空间、Harvard Dataverse，或者中国人民大学“清史数据共享平台”下载
- 请注意，`read_dta()`函数括号中的路径是编写者电脑文件的路径，请使用者根据自己电脑文件的路径进行修改，示例如下：

```
library(dplyr)
JSL1900_1912_clean <- tibble::as_tibble(JSL1900_1912_clean) # 转换数据集的类型
head(JSL1900_1912_clean,10) # 查看该数据集的前十行

library(haven)
Chushen_Recodes_1_ <- read_dta("/Users/jc/Documents/2-数据库-缙绅录数据库/原始文件/缙绅录公开版（新）/CGED-Q JSL Public Release Chushen Recodes.dta")
library(haven)
jiguansheng_Recodes <- read_dta("/Users/jc/Documents/2-数据库-缙绅录数据库/原始文件/缙绅录公开版（新）/CGED-Q JSL Public Release Province of Origin 籍貫省 Recodes.dta") # 更改为你自己的储存路径

JSL1900_1912_clean_1 <- JSL1900_1912[-kongbaiming,]
JSL1900_1912_clean_2 <- merge(JSL1900_1912_clean_1,
    Chushen_Recodes_1_, by = “出身一”, all.x = TRUE)
```

代码说明

- 以上代码中，`JSL1900_1912_clean_1`为剔除掉空白名的新数据集，同时保留`JSL1900_1912_clean`这个数据集，因为我们在后面还会用到，所以在新数据`JSL1900_1912_clean_1`这个数据集上匹配

- 代码运行之后，我们就创建了一个自动整理好“出身一”变量的数据集`JSL1900_1912_clean_2`

- 连接环节中
  - 主数据集是`JSL1900_1912_clean_1`
  - 副数据集是`Chushen_Recodes_1`
  - 公共列为“出身一”
  - 连接方式为左连接

- 简单来说，即主数据集和副数据集根据它们相同的列“出身一”所匹配而生成一个新数据集

检验结果

- 那么，如何检验这个数据集是否匹配成功？检验代码需要用到我们之前学过的一些知识，代码如下：

```
# 在新的数据集中生成qiren变量，
# 这个代码在"创建新变量(1)"中出现过，其含义是利用ifelse函数判别旗人
JSL1900_1912_clean_2$qiren <-
  ifelse((JSL1900_1912_clean$旗分!="" |
          JSL1900_1912_clean$身份二!="" |
          JSL1900_1912_clean$姓=="" ),1,0)

table(JSL1900_1912_clean_2$出身一,
      JSL1900_1912_clean_2$qiren)
          # 原始的"出身一"变量，比较杂乱
table(JSL1900_1912_clean_2$chushen_category,
      JSL1900_1912_clean_2$qiren)
          # 匹配之后生成的新的变量，较为统一

library(table1)
JSL1900_1912_clean_2$qiren[JSL1900_1912_clean_2$qiren == 1] <- "旗"
JSL1900_1912_clean_2$qiren[JSL1900_1912_clean_2$qiren == 0] <- "民”

table1(~chushen_category|qiren,
       data = JSL1900_1912_clean_2,
       overall = "total")
```

	旗 (N=154873)	民 (N=484559)	total (N=639432)
chushen_category			
吏員	1575 (1.0%)	1004 (0.2%)	2579 (0.4%)
武職出身	519 (0.3%)	1242 (0.3%)	1761 (0.3%)
生員	397 (0.3%)	7113 (1.5%)	7510 (1.2%)
監生	44013 (28.4%)	136014 (28.1%)	180027 (28.2%)
空白	31413 (20.3%)	100994 (20.8%)	132407 (20.7%)
翻譯生員	549 (0.4%)	4707 (1.0%)	5256 (0.8%)
翻譯舉人	606 (0.4%)	578 (0.1%)	1184 (0.2%)
翻譯進士	409 (0.3%)	397 (0.1%)	806 (0.1%)
舉人	26673 (17.2%)	73154 (15.1%)	99827 (15.6%)
蔭生	3081 (2.0%)	8922 (1.8%)	12003 (1.9%)
貢生	13398 (8.7%)	44143 (9.1%)	57541 (9.0%)
貢生 - 异途	17406 (11.2%)	57319 (11.8%)	74725 (11.7%)
進士	9023 (5.8%)	31097 (6.4%)	40120 (6.3%)
Missing	5811 (3.8%)	17875 (3.7%)	23686 (3.7%)

6.3.4 整理“籍贯省”变量

- 匹配完“出身一”之后，我们可以继续匹配“籍贯省”
- 需要注意的是，`merge()`函数一次只能匹配一个副数据集，所以我们需要在“出身一”匹配成功的基础上创建一个新的数据集来匹配“籍贯省”

```
JSL1900_1912_clean_3 <- merge(JSL1900_1912_clean_2,
                              jiguansheng_Recodes,
                              by = "籍贯省",
                              all= TRUE)

table(JSL1900_1912_clean_3$籍贯省, JSL1900_1912_clean_3$qiren)

table(JSL1900_1912_clean_3$籍贯省_clean, JSL1900_1912_clean_3$qiren)
```

制表检验结果

- 检验方式与之前相同:

```
JSL1900_1912_clean_3 <- filter(JSL1900_1912_clean_3,
                         !is.na(阳历年份) &
                          !is.na(chushen_category)&
                          !is.na(籍貫省_clean))
# 利用filter()函数清理掉含有NA的行

JSL1900_1912_clean_3$qiren[JSL1900_1912_clean_3$qiren == 1] <- "旗"
JSL1900_1912_clean_3$qiren[JSL1900_1912_clean_3$qiren == 0] <- "民”

table1(~籍貫省_clean|qiren,
      data = JSL1900_1912_clean_3,
      overall = "total")
```

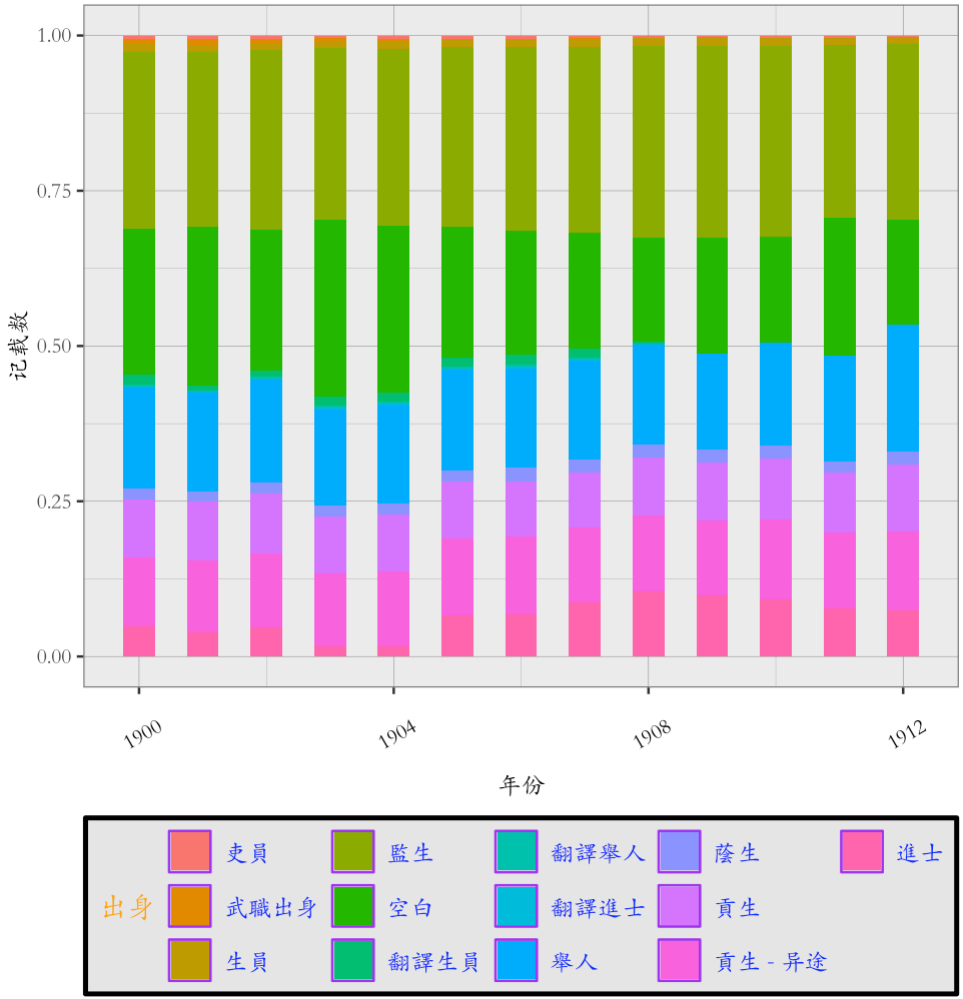
	旗 (N=149062)	民 (N=466684)	total (N=615746)
籍貫省_clean			
不顯	3 (0.0%)	19 (0.0%)	22 (0.0%)
其他	1 (0.0%)	1 (0.0%)	2 (0.0%)
四川	5424 (3.6%)	16005 (3.4%)	21429 (3.5%)
外國	130 (0.1%)	539 (0.1%)	669 (0.1%)
奉天	1028 (0.7%)	3130 (0.7%)	4158 (0.7%)
安徽	5878 (3.9%)	18622 (4.0%)	24500 (4.0%)
山東	4226 (2.8%)	13680 (2.9%)	17906 (2.9%)
山西	1789 (1.2%)	5619 (1.2%)	7408 (1.2%)
廣東	3579 (2.4%)	11768 (2.5%)	15347 (2.5%)
廣西	1575 (1.1%)	5139 (1.1%)	6714 (1.1%)
江南	23 (0.0%)	57 (0.0%)	80 (0.0%)
江蘇	9185 (6.2%)	28909 (6.2%)	38094 (6.2%)
江西	4163 (2.8%)	13637 (2.9%)	17800 (2.9%)
河南	3656 (2.5%)	11493 (2.5%)	15149 (2.5%)
浙江	12158 (8.2%)	37162 (8.0%)	49320 (8.0%)
湖北	3964 (2.7%)	12233 (2.6%)	16197 (2.6%)
湖南	5857 (3.9%)	18126 (3.9%)	23983 (3.9%)
甘肅	770 (0.5%)	2484 (0.5%)	3254 (0.5%)
直隸	4897 (3.3%)	15713 (3.4%)	20610 (3.3%)
福建	2921 (2.0%)	9070 (1.9%)	11991 (1.9%)
空白	65163 (43.7%)	204261 (43.8%)	269424 (43.8%)
貴州	1984 (1.3%)	5854 (1.3%)	7838 (1.3%)
陝西	1823 (1.2%)	5615 (1.2%)	7438 (1.2%)
雲南	1646 (1.1%)	4882 (1.0%)	6528 (1.1%)
順天	7219 (4.8%)	22666 (4.9%)	29885 (4.9%)

制图检验结果

```
# 已省略`theme()`函数
# 下面利用整理好的变量来绘图，
# 利用之前学习过的ggplot()函数来制作一个简单的直方图
library(ggplot2)
P_BarChart_Chushen <- ggplot(JSL1900_1912_clean_3, aes(阳历年份, fill =
chushen_category,)) +
  geom_bar(position = "fill",just= 0.5,width = 0.5) +
  labs(x = "年份",y = "记载数",title = "1900-1912年文官出身结构") +
  guides(fill=guide_legend(title = "出身"))

P_BarChart_Chushen # 显示图片
```

1900-1912年文官出身结构



图像说明

- 上图较为清晰地展示了1900-1912年文官出身结构的变化
- 可以看出，清末十年文官的出身结构比较稳定，文官主要来源为正途贡生、异途贡生、监生、进士、举人。
- `merge()`函数可用来整理数据库中记录类型比较多的变量，并且利用制表和制图函数还可以检验匹配效果
- 使用者还可以尝试整理“任职地”来制图制表，发现缙绅录中的诸多信息。

04

如何创建一个副数据集

6.4 创建一个副数据集以整理变量

- 以上我们匹配数据集的代码来自李康团队制作的代码，比较便利。但如果使用者需要利用到“出身一”、“籍贯省”之外的其他变量进行研究，该如何自行制作一个副数据集来整理变量呢？具体步骤如下：

  - 用`table()`函数将某个变量的所有数据显示出来
  - 复制`table()`的表格到Excel中
  - 利用excel进行分列，提出主要的数据信息
  - 对数据信息进行编辑，生成新的一列，整理的信息与原数据信息一一对应
  - 保存整理好的文件，导入到RStudio中
  - 利用`merge()`函数进行匹配

```
# 以整理“铨选方式”为例
table(JSL1900_1912_clean_3$铨选方式,
    JSL1900_1912_clean_3$qiren)
# 将`table()`的表格复制到Excel中进行
# 对应地编辑，并保存为数据集
install.packages("readxl")
library(readxl)
quanxuanfangshi_sort <- read_excel("/Users/jc/Documents/5-教学-R语言在缙绅录数据库中的运用v1.1.6/R语言代码和数据集/quanxuanfangshi_sort.xlsx") # 更改为你自己的储存路径
JSL1900_1912_clean_4 <- merge(JSL1900_1912_clean_3,
                              quanxuanfangshi_sort, by = "铨选方式",all= TRUE)
table(JSL1900_1912_clean_4$铨选方式_sort,JSL1900_1912_clean_4$qiren)
JSL1900_1912_clean_4$qiren[JSL1900_1912_clean_4$qiren == 1] <- "旗"
JSL1900_1912_clean_4$qiren[JSL1900_1912_clean_4$qiren == 0] <- "民"
table1(~铨选方式_sort | qiren, data = subset(JSL1900_1912_clean_4, !is.na(qiren)), overall = "total")
# 文官铨选方式统计
table1(~铨选方式_sort | qiren,
     data = subset(JSL1900_1912_clean_4, !is.na(qiren)), overall = "total")
```

注意事项

- 由于`JSL1900_1912_clean_4`数据集的`qiren`变量存在缺失值，故需要利用`subset()`函数在不改变原数据的情况下剔除掉`qiren`变量中的缺失值进行制表，不然RStudio会报错

- 当遇到`table1()`制表后RStudio返回列变量存在缺失值的错误指令时

  - 第一种方法是利用`data$var[is.na(var)] <- "text"`代码，即更改数据中缺失值为指定内容，但是这种方法需要更改原始数据，故不推荐

  - 第二种方法是在制表前，利用`filter()`函数筛除缺失值，具体请参照6.2.1中“检验籍贯省变量是否匹配成功”代码块的第一条代码

  - 第三种方法即是利用`subset()`函数在不改变原数据的情况下忽略掉`qiren`变量中的缺失值进行制表，这种方法简单快捷且保留了原始数据，故推荐用这种方法

代码运行结果

	旗 (N=149062)	民 (N=466684)	total (N=615746)
铨选方式_sort			
不详	43 (0.0%)	145 (0.0%)	188 (0.0%)
升	993 (0.7%)	2927 (0.6%)	3920 (0.6%)
差委	5 (0.0%)	4 (0.0%)	9 (0.0%)
授	675 (0.5%)	1919 (0.4%)	2594 (0.4%)
空白	61 (0.0%)	721 (0.2%)	782 (0.1%)
署	7 (0.0%)	20 (0.0%)	27 (0.0%)
补	15539 (10.4%)	47906 (10.3%)	63445 (10.3%)
袭	1 (0.0%)	9 (0.0%)	10 (0.0%)
调	2025 (1.4%)	6036 (1.3%)	8061 (1.3%)
选	17678 (11.9%)	53238 (11.4%)	70916 (11.5%)
题	32 (0.0%)	80 (0.0%)	112 (0.0%)
Missing	112003 (75.1%)	353679 (75.8%)	465682 (75.6%)

表格说明

- 观察上图可以发现，经过整理后的“铨选方式”变量变得更加简洁，更有利于分析
- 使用者可根据研究需要自行创建副数据集来整理变量。

05

连接两个数据集

6.5 连接两个数据集
`inner_join()`函数

- 在缙绅录的分析中，我们进行两个数据集的匹配，以追踪官员的仕途

- 比如，我们确定了1900年秋天在任的官员，但同时又想知道这些官员在1901年的秋天是否还在任，该怎么处理呢？我们需要用到连接代码`inner_join()`，格式如下：

```
`inner_join(x, y, by = , na_matches = ,)`
```

- `x`为主数据集，`y`为副数据集，`by`选定两个数据集中相同的列，与`merge`函数用法相似，如果选择多个列进行连接，则采用`by = c("","","",…)`的形式，`na_matches`表示是否匹配含NA的行，通常选择`never`

```
# 下面尝试创建连接只有1900年秋记录和只有1901年秋记录的数据集，考察官员一年内在任情况
#  准备数据
JSL1900_1912_clean %>%
        filter((年份季节 == 1900.5)) %>%
        select(阳历年份,地区,机构一,官职一,姓,名,字号,籍贯省,籍贯县,旗分,出身一,年份季节) ->
JSL1900fall
JSL1900_1912_clean %>%
        filter((年份季节 == 1901.5)) %>%
        select(阳历年份,地区,机构一,官职一,姓,名,字号,籍贯省,籍贯县,旗分,出身一,年份季节) ->
JSL1901fall

# 尝试用`inner_join()`函数将两个数据集连接起来，看看有多少官员1900秋-1901秋均在任
JSL1900_1901_inner_join <- inner_join(JSL1900fall,
                                      JSL1901fall,
                                      by = c("名","姓","字号","籍贯省","籍贯县","出身一","旗分"),
                                      keep = FALSE,
                                      na_matches = "never")
```

代码说明

- `by=c（…）` 表示公共列，这里我们用了集合的方式选出了7个公共列，这意味着两个数据集的行都需要满足7个条件才可进行连接

- `keep`表示是否保留合并之后副数据集的连接列，选择TRUE可以检验连接是否正常，选择FALSE可以简化数据集，仅凭个人意愿选择

- `na_matches = never`表示不匹配含有NA的行。新数据集`JSL1900_1901_inner_join`即为两个数据集连接后产生的新数据集，检验方式非常简单，连接后的数据集数量必须小于等于原两个数据集的和

- 使用者可根据研究需要连接匹配数据集

Thanks!