数据清洗

黄湘云

April 17, 2016

目录

数据清理实战

dplyr 数据清理

dplyr 基本函数

dplyr 高级函数

批量读取文件

处理日期格式

数据通常保存在.text .dat .csv .xlsx .xls 格式文件中，并且文件名与日期相关

```
# seq(from=as.Date("2015-01-01"),to=as.Date("2015-01-05"),by="day")
# seq(from=as.Date("2015-01-01"),to=as.Date("2015-12-31"),by="month")
# seq(from=as.Date("2015-03-31"),to=as.Date("2015-12-31"),by="quarter")
(mydate<-seq(from=as.Date("20160401","%Y%m%d"),
             to=as.Date("20160405","%Y%m%d"),by="day"))

## [1] "2016-04-01" "2016-04-02" "2016-04-03" "2016-04-04" "2016-04-05"

paste0(as.character(format(mydate,"%Y%m%d")),".xlsx") #delete "-"

## [1] "20160401.xlsx" "20160402.xlsx" "20160403.xlsx" "20160404.xlsx"
## [5] "20160405.xlsx"
```

批量读取文件

读入数据

```r
# faster than read.csv read.table
library(readxl) #xlsx xls
library(xlsx,quietly = TRUE) # xlsx need 32bit R
library(readr)  #.text .csv .dat
library(rio) # import export
library(XLConnect,quietly = TRUE) #Excel Connector for R
library(gdata) #Various R Programming Tools for Data Manipulation
# library(xlsReadWrite)
path<-"C:/Users/Xiangyun Huang/Desktop/CDA Task/input/"
setwd(path)
colNum <- 203 # ncol
rowNum <- 3951 # nrow
DataList<-list(NULL)
DataFrame<-as.data.frame(matrix(0,ncol = colNum,nrow = rowNum))
# data struct
for (i in seq(23)) {
  DataList[[i]] <- DataFrame
}
# read file
for (i in seq(length(mydate))) {
  file=paste0(mydate[i],".xlsx")
  DataList[[i]] <- read_excel(file,sheet = 1,col_names = TRUE,na="")
  # DataList[[i]] <- import(file) # call rio
}
```

数据库的导入

- MySQL: RMySQL
- SQL Server: RODBC
- SQLite: RSQLite

读取文本文件

```
read.table(file,
           header = FALSE,
           sep = "",
           nrows = -1,
           skip = 0,
           fill = !blank.lines.skip)
```

- file: 文件路径
- sep: 文件的分隔符
- skip: 跳过开始的 skip 行开始读取
- header: 是否将第一行读取的列名
- nrows: 读取的行数
- fill: 将缺失数据定为 NA

读取文本文件

```
read.table(file = "readtable/file1.txt",header = T,sep = "\t")
read.table(file = "readtable/file1.txt",header = T,sep = " ")
read.table(file = "readtable/file1.txt",header = T,sep = ",")
write.csv(chicagoNMMAPS,"chicagoNMMAPS.csv",na="NA",row.names = F)
```

目录

数据清理实战

dplyr 数据清理

dplyr 基本函数

dplyr 高级函数

R 中的数据清理

除了 R 中自带的各类函数，有各式各样的扩展包

数据导入

- foreign haven: 提供各类数据接口，如 SPSS,SAS,Stata 等
- RODBC RMySQL RPostgreSQL RSQLite: 数据库接口
- RCurl rvest: 网络爬虫包接口
- quantmod zoo xts: 金融数据获取，处理
- ggmap: 谷歌地图 API
- baidumap: 百度地图接口

数据处理

- data.table: 数据框的扩展
- reshape2: 长宽型数据的转换
- VIM: 缺失值插补和可视化
- plyr: 向量化数据处理
- caret: 挖掘前数据的处理(缺失值，训练集)
- plyrmr: 用于 RHadoop 的分布式存储数据的处理
- tidyr(数据整理) stringr(字符串操作)

dplyr: A Grammar of Data Manipulation

数据操作语法

专注于做类数据框的数据处理和汇总

- 内存内和内存外数据处理
- 更简单的语法
- 更快的速度

```
install.packages("dplyr")
library(dplyr)
```

dplyr

数据处理是数据挖掘中占用时间最长的工作

需要做的：

- 想清楚要做什么
- 在程序中精确地描述需求
- 调试、运行代码

dplyr 提供的：

- 阐明大部分常用的处理方法
- 提供简单的处理语法，便于快速编写 R 代码
- 基于 C++ 编写，快速完成计算

plyr: Tools for Splitting, Applying and Combining Data

数据拆分、应用、合并工具集

```
install.packages("plyr")
library(plyr)
```

dplyr 结构

- dplyr 基础函数
  - 筛选，子集选取，排序
- dplyr 高级函数
  - 汇总，连接，管道操作
- 其他有用的函数
  - each,colwise,do
- 现实数据处理案列

dplyr 对象 tbl 介绍

tbl 是 dplyr 定义的数据类型
可以接受：

- data.frame
- cube
- sql

```
chicagoNMMAPS<-read.csv(file = "chicagoNMMAPS.csv",
                        header = T,
                        sep = ",")
class(chicagoNMMAPS)
head(chicagoNMMAPS)
library(dplyr)
chicagoNMMAPS<-tbl_df(chicagoNMMAPS)
class(chicagoNMMAPS)
head(chicagoNMMAPS)
```

目录

数据清理实战

dplyr 数据清理

dplyr 基本函数

dplyr 高级函数

dplyr 主要操作

函数参数

- 第一个变量为数据框或者 tbl 对象
- 之后变量为筛选 (运算) 的条件
- 输出还是数据框

- 数据筛选
- 子集选取
- 数据排序
- 数据扩展

基本函数

- filter
- select
- arrange
- mutate
- summarise

基本函数–filter

```
library(dplyr)
df<-data.frame(
  color=c("blue","black","blue","blue","black"),
  value=1:5)
tbl<-tbl_df(df)
tbl
```

```
## Source: local data frame [5 x 2]
##
##    color value
##   (fctr) (int)
## 1   blue     1
## 2  black     2
## 3   blue     3
## 4   blue     4
## 5  black     5
```

基本函数–filter

```
filter(tbl,color=="blue")
```

```
## Source: local data frame [3 x 2]
##
##    color value
##   (fctr) (int)
## 1   blue     1
## 2   blue     3
## 3   blue     4
```

```
filter(tbl,value %in% c(1,4))
```

```
## Source: local data frame [2 x 2]
##
##    color value
##   (fctr) (int)
## 1   blue     1
## 2   blue     4
```

基本函数–filter

```
chicagoNMMAPS<-read.csv(file = "chicagoNMMAPS.csv",
                        header = T,
                        sep = ",")
chicagoNMMAPS<-tbl_df(chicagoNMMAPS)
filter(chicagoNMMAPS,dow=="Thursday")

## Source: local data frame [731 x 14]
##
##          date  time  year month   doy      dow death   cvd  resp
##        (fctr) (int) (int) (int) (int)   (fctr) (int) (int) (int)
## 1  1987-01-01     1  1987     1     1 Thursday   130    65    13
## 2  1987-01-08     8  1987     1     8 Thursday   109    51    13
## 3  1987-01-15    15  1987     1    15 Thursday   109    54    10
## 4  1987-01-22    22  1987     1    22 Thursday   121    60    10
## 5  1987-01-29    29  1987     1    29 Thursday   123    54     8
## 6  1987-02-05    36  1987     2    36 Thursday   127    71     7
## 7  1987-02-12    43  1987     2    43 Thursday   116    53    10
## 8  1987-02-19    50  1987     2    50 Thursday   115    48    15
## 9  1987-02-26    57  1987     2    57 Thursday   119    61    10
## 10 1987-03-05    64  1987     3    64 Thursday   125    66     9
## ..        ...   ...   ...   ...   ...      ...   ...   ...   ...
## Variables not shown: temp (dbl), dptp (dbl), rhum (dbl), pm10 (dbl), o3
##   (dbl)
```

基本函数–filter

```
filter(chicagoNMMAPS,year=="1987" & death < 100 & dow=="Thursday")
```

```
## Source: local data frame [3 x 14]
##
##         date  time  year month   doy      dow death   cvd  resp      temp
##       (fctr) (int) (int) (int) (int)   (fctr) (int) (int) (int)     (dbl)
## 1 1987-04-02    92  1987     4    92 Thursday    91    48     9 -1.388889
## 2 1987-06-11   162  1987     6   162 Thursday    96    45     3 22.500000
## 3 1987-07-30   211  1987     7   211 Thursday    92    50     5 26.388889
## Variables not shown: dptp (dbl), rhum (dbl), pm10 (dbl), o3 (dbl)
```

基本函数–select 数据筛选 子集选取

```
select(tbl,color)

## Source: local data frame [5 x 1]
##
##    color
##   (fctr)
## 1   blue
## 2  black
## 3   blue
## 4   blue
## 5  black

select(tbl,-color)

## Source: local data frame [5 x 1]
##
##   value
##   (int)
## 1     1
## 2     2
## 3     3
## 4     4
## 5     5
```

基本函数–select

用于选择的函数

- starts_with(x,ignore.case=TRUE)：以字符 x 开始的变量
- end_with(x,ignore.case=TRUE)：以字符 x 结束的变量
- contains(x,ignore.case=TRUE)：选取所有含字符 x 的变量
- matches(x,ignore.case=TRUE)：选取匹配正则表达式 x 的变量
- num_range("x",1:5,width=2)：选取 x01 到 x05 的变量
- one_of("x","y","z")：选取包含在声明变量中的
- everything()：选取所有变量

基本函数–select

```
names(chicagoNMMAPS)

##  [1] "date"  "time"  "year"  "month" "doy"   "dow"   "death" "cvd"
##  [9] "resp"  "temp"  "dptp"  "rhum"  "pm10"  "o3"

select(chicagoNMMAPS,death,pm10,rhum)

## Source: local data frame [5,114 x 3]
##
##    death     pm10   rhum
##    (int)    (dbl)  (dbl)
## 1    130 26.95607 95.500
## 2    150       NA 88.250
## 3    101 32.83869 89.500
## 4    135 39.95607 84.500
## 5    126       NA 74.500
## 6    130 40.95607 77.375
## 7    129 33.95607 74.500
## 8    109 28.95607 77.875
## 9    125 32.34877 95.125
## 10   153       NA 81.875
## ..   ...      ...    ...
```

基本函数–select

```
select(chicagoNMMAPS,starts_with("d"))

## Source: local data frame [5,114 x 5]
##
##          date   doy       dow death   dptp
##        (fctr) (int)    (fctr) (int)  (dbl)
## 1  1987-01-01     1  Thursday   130 31.500
## 2  1987-01-02     2    Friday   150 29.875
## 3  1987-01-03     3  Saturday   101 27.375
## 4  1987-01-04     4    Sunday   135 28.625
## 5  1987-01-05     5    Monday   126 28.875
## 6  1987-01-06     6   Tuesday   130 35.125
## 7  1987-01-07     7 Wednesday   129 26.750
## 8  1987-01-08     8  Thursday   109 22.000
## 9  1987-01-09     9    Friday   125 29.000
## 10 1987-01-10    10  Saturday   153 27.750
## ..        ...   ...       ...   ...    ...
```

基本函数–select

```
select(chicagoNMMAPS,contains('p'))

## Source: local data frame [5,114 x 4]
##
##     resp       temp   dptp     pm10
##    (int)      (dbl)  (dbl)    (dbl)
## 1     13 -0.2777778 31.500 26.95607
## 2     14  0.5555556 29.875       NA
## 3     11  0.5555556 27.375 32.83869
## 4      7 -1.6666667 28.625 39.95607
## 5     12  0.0000000 28.875       NA
## 6     12  4.4444444 35.125 40.95607
## 7     12  1.3888889 26.750 33.95607
## 8     13 -1.6666667 22.000 28.95607
## 9      7 -3.0555556 29.000 32.34877
## 10    11  0.2777778 27.750       NA
## ..   ...        ...    ...      ...
```

区别

filter& select

- filter: 通过一些准则来选取观测值 (行)
- select: 通过名字来选取变量 (列)

select& rename

- select: 只会选择变量，删除其余变量
- rename: 保留所有变量，可以用来改名

rename 函数

```
head(select(
  iris,plength=Petal.Length
  ))
```

```
##   plength
## 1     1.4
## 2     1.4
## 3     1.3
## 4     1.5
## 5     1.4
## 6     1.7
```

```
head(rename(
  iris,Plength=Petal.Length
  ))[,1:3]
```

```
##   Sepal.Length Sepal.Width Plength
## 1          5.1         3.5     1.4
## 2          4.9         3.0     1.4
## 3          4.7         3.2     1.3
## 4          4.6         3.1     1.5
## 5          5.0         3.6     1.4
## 6          5.4         3.9     1.7
```

数据排序–arrange

```
df_order<-order(tbl$color)
# tbl[df_order,]
arrange(tbl,color)

## Source: local data frame [5 x 2]
##
##    color value
##   (fctr) (int)
## 1  black     2
## 2  black     5
## 3   blue     1
## 4   blue     3
## 5   blue     4
```

数据排序–arrange

```
arrange(tbl,desc(color))
```

```
## Source: local data frame [5 x 2]
##
##    color value
##   (fctr) (int)
## 1   blue     1
## 2   blue     3
## 3   blue     4
## 4  black     2
## 5  black     5
```

数据排序–arrange

```
tb<-select(chicagoNMMAPS,date,death)
arrange(tb,date,desc(death))
```

```
## Source: local data frame [5,114 x 2]
##
##          date death
##        (fctr) (int)
## 1  1987-01-01   130
## 2  1987-01-02   150
## 3  1987-01-03   101
## 4  1987-01-04   135
## 5  1987-01-05   126
## 6  1987-01-06   130
## 7  1987-01-07   129
## 8  1987-01-08   109
## 9  1987-01-09   125
## 10 1987-01-10   153
## ..        ...   ...
```

数据扩展–mutate

在保留原变量的基础上增加变量

```
mutate(tbl,double=2*value)
```

```
## Source: local data frame [5 x 3]
##
##    color value double
##   (fctr) (int)  (dbl)
## 1   blue     1      2
## 2  black     2      4
## 3   blue     3      6
## 4   blue     4      8
## 5  black     5     10
```

```
# mutate(tbl,double=2*value,quad=4*value)
```

从字符串提取年月日

常常需要从原始数据中扩展日期，以便分年、月、日分析数据

```
(year<-substr("1999-01-01",1,4))

## [1] "1999"

(month<-substr("1999-01-01",6,7))

## [1] "01"

(day<-substr("1999-01-01",9,10))

## [1] "01"
```

原始数据中有一列日期数据比如 1999-01-01 至 2016-04-17，列名为 date，则扩展三列只需用 date 替换下面的 1999-01-01

```
mutate(mydata,year=substr("1999-01-01",1,4),
       month=substr("1999-01-01",6,7),
       day=substr("1999-01-01",9,10))
```

数据扩展–transmute

transmute 函数与 mutate 类似，但是它会删除原有的变量

```
transmute(tbl,double=2*value,quad=4*value)
```

```
## Source: local data frame [5 x 2]
##
##   double  quad
##    (dbl) (dbl)
## 1      2     4
## 2      4     8
## 3      6    12
## 4      8    16
## 5     10    20
```

数据汇总–summarise

summarise: 将多个值汇总为一个数据值

```
summarise(tbl,total=sum(value),avg=mean(value))
```

```
## Source: local data frame [1 x 2]
##
##   total   avg
##   (int) (dbl)
## 1    15     3
```

```
sum(tbl$value)
```

```
## [1] 15
```

数据汇总–summarise

汇总函数

在 summarise() 中使用的汇总函数，要求输入参数是一个向量，返回是一个值。
R 自带的统计函数都是可以的：
min(),max(),mean(),sum(),sd(),median(),IQR().
此外，dplyr 还提供了一些其他会用到的函数：
n()：观测值的个数
n_distinct(x)：不同观测值的个数
first(x),last(x) 和 nth(x,n) 获取第一个、最后一个和第 n 个数据

```
summarise(chicagoNMMAPS,first=first(date),last=last(date))
```

```
## Source: local data frame [1 x 2]
##
##        first       last
##       (fctr)     (fctr)
## 1 1987-01-01 2000-12-31
```

dplyr 基本函数

summary 小结

- filter: 根据准则选择观测值
- select: 根据名字选择变量
- arrange: 对观测值进行排序
- mutate: 增添新的观测值
- summarise: 汇总数据

目录

数据清理实战

dplyr 数据清理

dplyr 基本函数

dplyr 高级函数

dplyr 高级函数

- 数据集的连接 join
- 分组汇总 group_by
- 管道函数%>%
- 其他函数 do 和 plyr::colwise
- MySQL 数据库的连接

数据连接–join

如何将两个数据集连接起来?

- dat1

```
##      name instrument
## 1    John     guitar
## 2    Paul       bass
## 3  George     guitar
## 4   Ringo      drums
## 5  Stuart       bass
## 6    Peta      drums
```

- dat2

```
##      name band
## 1    John    T
## 2    Paul    T
## 3  George    T
## 4   Ringo    T
## 5   Brian    F
```

left_join 函数

保留缺失值所在行

```
left_join(dat1,dat2,by="name")

## Warning in left_join_impl(x, y, by$x, by$y):
joining factors with different levels, coercing to
character vector

##     name instrument band
## 1   John     guitar    T
## 2   Paul       bass    T
## 3 George     guitar    T
## 4  Ringo      drums    T
## 5 Stuart       bass <NA>
## 6   Peta      drums <NA>
```

实际中这个函数是用的最多的（相比较后面三个）

inner_join 函数

去掉缺失值所在行

```
inner_join(dat1,dat2,by="name")
```

```
## Warning in inner_join_impl(x, y, by$x, by$y):
joining factors with different levels, coercing to
character vector
```

```
##     name instrument band
## 1   John     guitar    T
## 2   Paul       bass    T
## 3 George     guitar    T
## 4  Ringo      drums    T
```

semi_join 函数

以 dat1 中的 name 变量为对照标准，取匹配到的观测 name

```
semi_join(dat1,dat2,by="name")
```

```
## Warning in semi_join_impl(x, y, by$x, by$y):
joining factors with different levels, coercing to
character vector
```

```
##     name instrument
## 1   John     guitar
## 2   Paul       bass
## 3 George     guitar
## 4  Ringo      drums
```

anti_join 函数

与 semi_join 相反以 dat1 中的 name 变量为对照标准，取没有匹配到的观测 name

```
anti_join(dat1,dat2,by="name")
```

```
## Warning in anti_join_impl(x, y, by$x, by$y):
joining factors with different levels, coercing to
character vector
```

```
##     name instrument
## 1   Peta      drums
## 2 Stuart       bass
```

分类汇总 – group_by

```
summarise(tbl,total=sum(value))

## Source: local data frame [1 x 1]
##
##   total
##   (int)
## 1    15

by_color<-group_by(tbl,color) ## group_by
summarise(by_color,total=sum(value))

## Source: local data frame [2 x 2]
##
##    color total
##   (fctr) (int)
## 1  black     7
## 2   blue     8
```

分类汇总 – group_by

```
chicago_year<-group_by(chicagoNMMAPS,year)
summarise(chicago_year,year_death=sum(death),
          average_death=mean(death),max(month))
```

```
## Source: local data frame [14 x 4]
##
##     year year_death average_death max(month)
##    (int)      (int)         (dbl)      (int)
## 1   1987      42583      116.6658         12
## 2   1988      43469      118.7678         12
## 3   1989      42981      117.7562         12
## 4   1990      42313      115.9260         12
## 5   1991      42640      116.8219         12
## 6   1992      41714      113.9727         12
## 7   1993      43449      119.0384         12
## 8   1994      42937      117.6356         12
## 9   1995      43806      120.0164         12
## 10  1996      41784      114.1639         12
## 11  1997      40340      110.5205         12
## 12  1998      40101      109.8658         12
## 13  1999      41776      114.4548         12
## 14  2000      40359      110.2705         12
```

管道函数

%>% 或者%.% 可以将上一个函数的输出作为下一个函数的输入:

```
1:5 %>% mean()
```

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

```
1:5 %>% mean(.) %>% sqrt()
```

```
## [1] 1.732051
```

```
# 1:5 %>% mean() %>% sqrt()
```

管道函数

- ```
  tem<-group_by(chicagoNMMAPS,month)
  summarise(tem,count=n())
  ```

```
Source: local data frame [12 x 2]

   month count
   (int) (int)
1      1   434
2      2   396
3      3   434
4      4   420
5      5   434
6      6   420
7      7   434
8      8   434
9      9   420
10    10   434
11    11   420
12    12   434
```

- ```
  chicagoNMMAPS %>%
    group_by(month) %>%
    summarise(count=n())
  ```

```
Source: local data frame [12 x 2]

   month count
   (int) (int)
1      1   434
2      2   396
3      3   434
4      4   420
5      5   434
6      6   420
7      7   434
8      8   434
9      9   420
10    10   434
11    11   420
12    12   434
```

colwise 和 do 函数

筛选每年最大的记录

```
library(dplyr)
order<-read.csv("order.csv")
order<- order %>% select(orderdate,totalprice) %>%
  mutate(year=substr(orderdate,1,4))
order %>% group_by(year) %>%
  summarise(max(totalprice))
```

do

筛选每年最大的前两条记录

do(data,fun(.))

```
order %>% group_by(year) %>%
  arrange(desc(totalprice)) %>%
  do(.,head(.,2))
```

colwise

自动对每一列调用函数

colwise(function)(data.frame)

```
library(plyr)
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

```
colwise(round)(iris[,1:4]) %>% head
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1            5           4            1           0
## 2            5           3            1           0
## 3            5           3            1           0
## 4            5           3            2           0
## 5            5           4            1           0
## 6            5           4            2           0
```

连接 MySQL

```
src<-src_mysql("sqlbook") # 连接数据库
orderSQL<-tbl(src,from="orders") # 调用数据库中的表
newOrder<- select(orderSQL,orderis,orderdate) #使用dplyr包中各个函数
newOrder
write.csv(newOrder,"newOrder.csv")
newOrder$query
```