# data.table 与 pandas | 统计之都

**cosx.org**/2021/01/dt-pd

数据分析项目通常可以分解为以下过程，数据加载 - 数据清洗 -(特征处理、可视化、模型训练)- 成果汇报 [1]。其中，数据清洗与特征处理或者称为数据预处理过程，一般会占据整个项目的大部分时间。熟练掌握相关工具，提高数据处理的效率，是开展数据分析工作的基础。

在开展数据科学相关工作时，最常用的开源工具包括 R 与 python。对于可在内存级处理的数据，在 R 中通常使用 data.table 包进行数据处理，而在 python 环境中 pandas 包最为常用的。为了方便查阅和对比，本文分别用 data.table 与 pandas 实现了常见的数据处理任务 [2]。

数据框（data frame）是大家接触最多的数据格式，它的每一列都是长度相等、类型一致的向量。对数据框的操作可以从行与列两个维度，拆解为以下五类基本操作。这一思路来自 dplyr 包 [3] 的帮助文档，因此下面五类基本操作的英文均为该包的函数名。这些基本操作均可以与 group_by 相互结合使用。除了这五类基本操作，还包括行列转换、数据框的切割与合并等。绝多数的数据处理任务都可以拆解为以上这几类基本操作，具体案例请参见下面的代码。

- 行：选择 filter、排序 arrange
- 列：选择 select、新建 mutate、计算 summarise

## 数据探索

### 数据加载

```
library(data.table)
packageVersion('data.table')

url =
"https://vincentarelbundock.github.io/Rdatasets/csv/datasets/HairEyeColor.csv"
dt = fread(url)
```

```
import pandas as pd
pd.__version__

url =
"https://vincentarelbundock.github.io/Rdatasets/csv/datasets/HairEyeColor.csv"
df = pd.read_csv(url)
```

## 查看数据结构

```
# 数据类型
class(dt)
str(dt)

# 列名
names(dt)

# 打印前后几行
head(dt, n=3)
tail(dt, n=3)

# 维度
dim(dt)
nrow(dt)
ncol(dt)

# 统计描述
summary(dt)
```

```
# 数据类型
type(df)
df.dtypes

# 列名
list(df)

# 打印前后几行
df.head(n=3)
df.tail(n=3)

# 维度
df.shape
len(df.index)
len(df.columns)

# 统计描述
df.describe()
```

## 行选择与排序

### 行选择

```
# 基于行所在位置筛选，data.table格式的index默认为1开始且
dt[c(3,1,2)]


# 单条件筛选
dt[Hair == 'Red']

# 多条件筛选
dt[Hair == 'Black' &
   Freq >= 10 &
   Eye %in% c('Brown', 'Blue')]

# 基于行所在位置筛选
df.iloc[[2,0,1]] # python序数从0开始，2代表第三行
df.loc[[2,0,1]] # 如果index未修改，效果与iloc的一致

# 单条件筛选，去掉.loc效果一致
df.loc[df['Hair'] == 'Red']

# pandas 多条件筛选时要用 |, &, ~分别代表or, and, not; 且每个条件需要用括号区分
df.loc[(df['Hair'] == 'Black') &
       (df['Freq'] >= 10) &
       (df['Eye'].isin(['Brown', 'Blue']))]
```

### 行排序

```
dt[order(Sex, -Freq)]

df.sort_values(['Sex', 'Freq'],
               ascending = [True, False] )
```

## 列选择、新建与计算

### 列选择

```
dt[, .(Hair, Freq)]
# or
dt[, c('Eye', 'Sex'), with=FALSE]

df[['Hair', 'Freq']]
# or
df.loc[:, ['Eye', 'Sex']] # 选一列时也要保留[]，否则与df.Eye一样为series
```

## 列新建

```
# 新建一列
dt[, nc := .I] # .I .N .SD为特殊符号,查看帮助?`.I`
dt[,'nc0'] = 1:32

# 新建多列
dt[, `:=`(
  nc1 = 1:32,
  nc2 = paste(Hair, Eye, sep=',')
)]

# 基于条件新建列
dt[, nc3 := ifelse(Freq >= 10, 1, 0)]
dt[Freq >= 20, nc4 := 2]

# 基于函数新建多列
ncols = c('nc', 'nc0')
dt[,
   (ncols) := lapply(.SD, function(x) x^0.5+1),
   .SDcols = ncols]

# 删除一列
dt[, nc := NULL]
# 删除多列
dt[, (c('nc0','nc1','nc2','nc3','nc4')) := NULL]
```

```
# 新建一列
df = df.assign(nc = pd.Series(range(32)))
df.loc[:,'nc0'] = pd.Series(range(32), index=df.index)

# 新建多列
df = df.assign(
  nc1 = pd.Series(range(32)),
  nc2 = df.Hair + ',' + df.Eye
)

# 基于条件新建列
df = df.assign(nc3 = df.Freq.apply(lambda x: 1 if x >= 10 else 0))
df.loc[df.Freq >= 20, 'nc4'] = 2

# 基于函数新建多列
ncols = ['nc', 'nc0']
df.loc[:, ncols] = df[ncols].apply(lambda x: x**0.5+1)

# 删除一列
df = df.drop('nc', axis=1)
# 删除多列
df.drop(['nc0','nc1','nc2','nc3','nc4'], axis=1, inplace=True)
```

## 列计算

```
# 对一列进行计算
dt[, max(Freq)] # 最大值
dt[, unique(Eye)] # 唯一值
dt[, table(Eye)] # 计数

# 对多列进行计算
## 所有列的最大值
dt[, lapply(.SD, max)]

## 所有列的缺失率
dt[, lapply(.SD, function(x) mean(is.na(x)))]

## 对部分列计算缺失率，且可扩展到其他函数
sel_cols = c('Hair', 'Sex', 'Freq')
dt[, lapply(.SD, function(x) mean(is.na(x))), .SDcols = sel_cols]

# 对一列进行计算
df.Freq.max() # 最大值
df.Eye.unique() # 唯一值
df.Eye.value_counts() # 计数

# 对多列进行计算
## 所有列的最大值
df.max()

## 所有列的缺失率
df.isnull().mean()

## 对部分列计算缺失率，且可扩展到其他函数
sel_cols = ['Hair', 'Sex', 'Freq']
df[sel_cols].apply(lambda x: x.isnull().mean())
```

## 分组数据操作

```
# 分组行操作
## 行选择
dt[, .SD[1], by = 'Sex'] # 每组的第一行
dt[, .SD[.N], by = 'Sex'] # 每组的最后一行


# 分组列操作
## 分组列新建
dt[, freq_total := sum(Freq), by = 'Sex']

## 分组列计算
dt[, .(freq_total = sum(Freq)), by = 'Sex'][]
```

```
# 分组行操作
## 行选择
df.groupby('Sex').head(1) # 每组的第一行
df.groupby('Sex').tail(1) # 每组的最后一行


# 分组列操作
## 分组列新建
df.loc[:,'freq_total'] = df.groupby('Sex')['Freq'].transform(sum)

## 分组列计算
df.groupby('Sex').agg({'Freq':'sum'}) \
  .rename(columns={'Freq':'freq_total'}) \
  .reset_index()
```

## 行列转换

### 长宽表转换

```
# 长表转宽表
dt_w = dcast(dt, Hair+Sex~Eye, value.var = 'Freq', fun.aggregate = sum)

# 宽表转长表
dt_l = melt(dt_w, id = c('Hair','Sex'), variable.name = 'Eye', value.name =
'Freq')

# 长表转宽表
df_w = pd.pivot_table(df, index=['Hair','Sex'], columns='Eye', values='Freq',
aggfunc = sum).reset_index()

# 宽表转长表
df_l = pd.melt(df_w, id_vars = ['Hair','Sex'], var_name='Freq')
```

### 行列切割合并

```
# 一行切割为多行
dtr = dt[, paste0(Eye, collapse = ','), keyby = c('Hair', 'Sex')]
dtr[, .(Eye = unlist(strsplit(V1, ','))), by = c('Hair', 'Sex')]

# 一列切割为多列
dtc = dt[, .(Hair, eye_sex = paste(Eye, Sex, sep = ','))]
dtc[, c('Eye', 'Sex') := tstrsplit(eye_sex, ',')][]

# 一行切割为多行
dfr = df.groupby(['Hair','Sex'])['Eye'].apply(lambda x: ','.join(x)).reset_index()
dfr.assign(Eye = dfr['Eye'].str.split(',')).explode('Eye')

# 一列切割为多列
dfc = df[['Hair']].assign(eye_sex = df.Eye+','+df.Sex)
dfc[['Eye', 'Sex']]= dfc['eye_sex'].str.split(',', expand = True)
```

## 多个数据框合并

### 数据框行合并

```
# 数据框行切割
dtlist1 = split(dt, by = 'Sex')
# or
dtlist2 = split(dt, list(dt$Sex))

# 数据框行合并
dtbind2 = rbindlist(dtlist1)

# 数据框行切割
dfdict = dict(tuple(df.groupby(['Sex'])))
# or
dflist = [d for _, d in df.groupby(['Sex'])]

# 数据框行合并
df_con = pd.concat(dfdict, axis=0).reset_index(drop=True)
```

### 数据框列合并

```
dt1 = dt[sample(.N,2)][,V1 := NULL]
dt2 = dt[sample(.N,3)][,V1 := NULL]
dt3 = dt[sample(.N,4)][,V1 := NULL]

# 合并两个数据框
dtmerge2 = merge(
  dt1, dt2,
  by = c('Hair', 'Eye', 'Sex'),
  all = TRUE
)
# all, all.x, all.y: TRUE, FALSE

# 合并多个数据框
dtmerge3 = Reduce(
  function(x,y) merge(
    x,y,
    by = c('Hair', 'Eye', 'Sex'),
    all = TRUE
  ),
  list(dt1, dt2, dt3)
)
```

```
df1 = df.sample(n=2).drop('Unnamed: 0', axis=1)
df2 = df.sample(n=3).drop('Unnamed: 0', axis=1)
df3 = df.sample(n=4).drop('Unnamed: 0', axis=1)

# 合并两个数据框
dfmerge2 = pd.merge(
  df1, df2,
  on = ['Hair', 'Eye', 'Sex'],
  how = 'outer'
)
# how: left, right, inner, outer

# 合并多个数据框
from functools import reduce
df_merge3 = reduce(
  lambda x,y: pd.merge(
    x,y,
    on = ['Hair', 'Eye', 'Sex'],
    how = 'outer'
  ),
  [df1, df2, df3]
)
```

## 总结

通过以上的对比介绍，大家可以从功能上直观地了解了，如何分别使用 data.table 和 pandas 实现常见数据分析任务。如果您希望更进一步了解这两个包的功能，请查看各自项目主页（data.table, pandas）。在性能方面的对比，根据 Database-like ops benchmark 显示，data.table 在大部分数据操作任务中性能表现最好，而且其语法也相对简洁统一。

---

1. R for Data Science↩
2. 本文参考了 Data Manipulation with Python Pandas and R Data.Table 并结合了自己的数据分析经验↩
3. dplyr 是 R 语言中另外一个广泛使用的数据处理工具包，其与 data.table 的对比请参考 A data.table and dplyr tour↩