复习与提示
DataFrame
R 数据类型
R 程序结构
R DataFrame
关系代数基本操作
关系代数制图
宽表到长表
dbplyr

DataFrame

续本达

清华大学 工程物理系

2024-07-26 清华

安装软件 R 的 ggplot2 与 dbplyr

```
apt install r-cran-ggplot2 r-cran-dbplyr r-cran-dplyr r-cran-reshape2 r-cran-rsqlit
```

最佳工具

- Dataframe 流派的关系代数系统 全局最佳工具是 GNU R
  - Python 只是小圈子的最佳工具
- Grammar of graphics 的全局最佳工具是 R ggplot2
  - seaborn 只是眼前的妥协，未来 Python 小圈子的最佳工具可能是 plotnine

关系代数复习

- 集合运算：$\cap, \cup, \setminus$
  - SQL `INTERSECT` `UNION` `EXCEPT`
- 线性运算：$\otimes, \Pi, \sigma$
  - SQL `SELECT` `WHERE`
- 关系运算：连接 $\bowtie$
  - SQL `JOIN`
- 拓展运算：GroupBy（分组）$\mathcal{G}$
  - SQL `GROUP BY`

[66%] 少年爱迪生想

- ☒ 比较物理系和工物系的男女比例
- ☒ 算出大家的总评成绩：
  - ☒ 小作业权相等，总体占 65% 的成绩
  - ☒ 大作业占 30% 的成绩
  - ☒ 划分出不同的分数段，给出某分数段同学的手机号
- ☐ 画出各班平均小作业成绩的变化曲线

```
SELECT 学号, SUM(权重*分数) AS 总评 FROM
(SELECT 学号, CASE 大作业
  WHEN 0 THEN 0.65
  WHEN 1 THEN 0.3
  END AS 权重, AVG(分数) AS 分数
  FROM longscores GROUP BY 学号, 大作业)
GROUP BY 学号
HAVING 总评 > 105
```

学号	总评
49	105.16375
55	105.70625

DataFrame

由 R 语言提出的数据表格形式：与 SQL 一样都脱胎于关系代数思想

SQL 的区别

- SQL 是描述性语言，与函数式编程更接近。
  - 生成的表格不能随意改动。改动只是通过创建新表格实现。
- DataFrame 是动态结构
  - 可以更灵活地更改；
  - 但过于灵活可能带来调试上的麻烦。
- SQL 可以处理大于内存容量的数据，DataFrame 必须借助 Spark 等大数据平台才能实现。

GNU R 语言

- R 是著名的统计学语言，应用广泛
- 起源于 1970 年代的 S 语言
  - 在 fortran 库的基础上交互地处理数据进行统计分析
    调用 fortran 库提供交互环境是 S, Matlab 和 Numpy 的共同起源
  - 提供交互式的绘图工具，以便于快速迭代
  - 受 LISP 语言的影响
  - DataFrame 概念影响了几乎所有统计工具
    例如 S-PLUS, SAS, SPSS 应用广泛
- GNU R 重新使用 Scheme 实现了 S 语言，并以自由软件形式发布
  - 自由软件和开源运动是为科学研究提供了可 复现 的工具

为什么使用 R

- 大部分统计学算法都有 R 语言的实现
  站在数理统计学天才们的肩膀上。
- 非常适合快速试验并找到统计方法上的 最佳工具

R 的重要数据处理模块

`dplyr` 数据整理的利器，简洁的语法表达关系代数

`dbplyr` 与 SQL 浑然连成一体，统一 `DataFrame` 与 SQL

`ggplot2` 科学制图利器，数据制图易用性优于 `matplotlib`

参考资料

1. Wickham, Çetinkaya-Rundel and Grolemund. R for Data Science: Import, Tidy, Transform, Visualize, and Model Data. 2nd edition.
2. 李东风 R 语言教程

数据类型

numeric 数值型：分成整型、浮点型和复数型

- 错误：numeric 就是 double ，不包括整型。

character 字符型

factor 枚举型

logical 布尔型

```
# 获得帮助
help(help)
help(numeric)
```

```
10 + 66
```

```
76
```

```
class(76)
```

```
numeric
```

数值的类型

class 与 typeof 的区别是什么?

```
typeof(76)
```

```
double
```

```
typeof(as.integer(76))
```

```
integer
```

numeric 包括整型与浮点型

```
class(1.3)
```

```
numeric
```

```
(53.2+5) * 2
```

```
116.4
```

```
2^8
```

```
256
```

向量

使用 c 命令构造向量

```
help(c)
```

```
c(3,2,1)
```

3
2
1

```
c(3,2,1) + c(4,7,8)
```

7
9
9

向量的数据类型

```r
c(3, "a")
```

```
3
a
```

```r
typeof(c(3, "a"))
```

```
character
```

变量

```
a <- c(3,2,1)
```

3
2
1

```
c(4,7,9) -> b
```

4
7
9

向量的运算

```
a^2 + 7
```

```
16
11
 8
```

向量永远是一维的

```
c(c(2,3), c(0,1))
```

2
3
0
1

```
c(c(2,3), 4)
```

2
3
4

多维数组

- 需要用 `array`

向量的索引

- R 的向量索引从 1 开始。

```
a[3]
```

```
1
```

```
a[2]
```

```
2
```

- 闭区间。非 Python 的左闭右开。

```
a[2:3]
```

```
2
1
```

向量作为索引

```
a[c(3,3,3,1,2)]
```

1
1
1
3
2

所有数字都是向量

```
length(a)
```

```
3
```

```
class(a)
```

```
numeric
```

```
length(1)
```

```
1
```

字符型

```
"Hello World!"
```

```
Hello World!
```

```
class("Hello World!")
```

```
character
```

- 字符型的向量

```
class(c("Hello", "World!"))
```

```
character
```

字符串操作

```
greek <- c("alpha", "beta", "gamma", "delta")
length(greek)
```

```
4
```

取子字符串

```
substr(greek, 2, 3)
```

```
lp
et
am
el
```

```
gsub('^a', 'A', greek)
```

```
Alpha
beta
gamma
delta
```

```
gsub('a', 'A', greek)
```

```
AlphA
betA
gAmmA
deltA
```

```
gsub('a$', 'A', greek)
```

```
alphA
betA
gammA
deltA
```

寻找子串

- grep 返回匹配的索引

```
grep('e', greek)
```

2
4

```
greek[grep('e', greek)]
```

beta
delta

布尔型

```r
class(TRUE)
```

```
logical
```

```r
TRUE & FALSE
```

```
FALSE
```

```r
!FALSE | FALSE
```

```
TRUE
```

```r
greek == "alpha"
```

```
[1]  TRUE FALSE FALSE FALSE
```

数值型

```
typeof(NaN)
```

```
double
```

```
1 + NaN
```

```
Inf + 1
```

```
Inf
```

```
Inf - Inf
```

枚举型

```
people <- c('男', '男', '男','女',
            '女', '男', '女', '女')
```

男
男
男
女
女
男
女
女

```
p <- as.factor(people)
```

男
男
男
女
女
男
女
女

枚举型的机制

```
as.numeric(p)
```

```
2
2
2
1
1
2
1
1
```

```
levels(p)
```

```
女
男
```

```
levels(p) <- c("woman", "man")
p
```

```
man
man
man
woman
woman
man
woman
woman
```

程序结构

选择结构

```
g <- 4
result <- if (g %% 2) {
  "奇数"
} else {
  "偶数"
}
```

偶数

```
rep(result, 3)
```

偶数
偶数
偶数

循环结构

```
for (i in 1:5) {
  print(i)
}
```

正态分布

- d 代表概率密度函数

```
dnorm(0, mean=0, sd=1)
```

```
0.398942280401433
```

- p 代表累积分布函数

```
pnorm(3) - pnorm(-3)
```

```
0.99730020393674
```

- q 代表分位数

```
qnorm(0.5)
```

```
0
```

其它概率分布

unif	均匀
pois	泊松
binom	二项
gamma	伽马
exp	指数

```
dunif(0.5)
```

```
1
```

```
dunif(2)
```

```
0
```

分布的例子

```
runif(5)
```

0.141196223907173
0.797878413693979
0.887777165044099
0.0513732228428125
0.987073467345908

```
rpois(5, 0.5)
```

0
2
0
1
0

DataFrame

```r
data(mtcars)
head(mtcars)
```

```
                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

类型

```
typeof(mtcars)
```

```
list
```

```
class(mtcars)
```

```
data.frame
```

```
typeof(mtcars$hp)
```

```
double
```

行的名字

```
head(row.names(mtcars))
```

```
Mazda RX4
Mazda RX4 Wag
Datsun 710
Hornet 4 Drive
Hornet Sportabout
Valiant
```

```
typeof(row.names(mtcars))
```

```
character
```

列的名字

```
names(mtcars)
```

mpg
cyl
disp
hp
drat
wt
qsec
vs
am
gear
carb

基本的统计量

```
sd(mtcars$mpg)
```

```
6.0269480520891
```

```
median(mtcars$mpg)
```

```
19.2
```

```
min(mtcars$mpg)
```

```
10.4
```

```
mean(mtcars$mpg)
```

```
20.090625
```

summary 命令

```
summary(mtcars$mpg)
```

```
 Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
10.40   15.43   19.20   20.09   22.80   33.90 
```

DataFrame 的 summary

```
summary(mtcars)
```

```
     mpg             cyl             disp             hp       
 Min.   :10.40   Min.   :4.000   Min.   : 71.1   Min.   : 52.0  
 1st Qu.:15.43   1st Qu.:4.000   1st Qu.:120.8   1st Qu.: 96.5  
 Median :19.20   Median :6.000   Median :196.3   Median :123.0  
 Mean   :20.09   Mean   :6.188   Mean   :230.7   Mean   :146.7  
 3rd Qu.:22.80   3rd Qu.:8.000   3rd Qu.:326.0   3rd Qu.:180.0  
 Max.   :33.90   Max.   :8.000   Max.   :472.0   Max.   :335.0  
     drat             wt             qsec             vs        
 Min.   :2.760   Min.   :1.513   Min.   :14.50   Min.   :0.0000  
 1st Qu.:3.080   1st Qu.:2.581   1st Qu.:16.89   1st Qu.:0.0000  
 Median :3.695   Median :3.325   Median :17.71   Median :0.0000  
 Mean   :3.597   Mean   :3.217   Mean   :17.85   Mean   :0.4375  
 3rd Qu.:3.920   3rd Qu.:3.610   3rd Qu.:18.90   3rd Qu.:1.0000  
 Max.   :4.930   Max.   :5.424   Max.   :22.90   Max.   :1.0000  
       am              gear            carb      
 Min.   :0.0000   Min.   :3.000   Min.   :1.000  
 1st Qu.:0.0000   1st Qu.:3.000   1st Qu.:2.000  
 Median :0.0000   Median :4.000   Median :2.000  
 Mean   :0.4062   Mean   :3.688   Mean   :2.812  
 3rd Qu.:1.0000   3rd Qu.:4.000   3rd Qu.:4.000  
 Max.   :1.0000   Max.   :5.000   Max.   :8.000  
```

DataFrame 用法

- 简单制图
- 线性回归
- 枚举类型

构造 data.frame

```
A <- data.frame(ID=c(1,2), name=c("Wang", "Li"))
```

1	Wang
2	Li

```
B <- data.frame(ID=c(2,3), name=c("Li", "Zhang"))
```

2	Li
3	Zhang

集合基本运算：并

```
AB <- rbind(A, B)
```

1	Wang
2	Li
2	Li
3	Zhang

```
unique(AB)
```

1	Wang
2	Li
3	Zhang

交与差

交

```
merge(A, B)
```

2 Li

差

```
library(dplyr)
setdiff(B, A)
```

3 Zhang

线性代数运算：笛卡尔积

```
AxB <- merge(A, B, by=NULL)
```

1	Wang	2	Li
2	Li	2	Li
1	Wang	3	Zhang
2	Li	3	Zhang

投影与选择

投影

```
A$name
```

```
Wang
Li
```

选择

```
subset(A, name=="Li")
```

```
2    Li
```

动态的 DataFrame

```
A$GPA = c(3, 4)
A
```

```
  ID name GPA
1  1 Wang   3
2  2   Li   4
```

```
B$age = c(18, 20)
B
```

```
  ID  name age
1  2    Li  18
2  3 Zhang  20
```

连接

```
merge(A, B)
```

```
  ID name GPA age
1  2   Li   4  18
```

左连接

```
merge(A, B, all.x=TRUE)
```

```
  ID name GPA age
1  1 Wang   3  NA
2  2   Li   4  18
```

连接（续）

右连接

```
merge(A, B, all.y=TRUE)
```

```
  ID  name GPA age
1  2    Li   4  18
2  3 Zhang  NA  20
```

外连接

```
merge(A, B, all=TRUE)
```

1	Wang	3	
2	Li	4	18
3	Zhang		20

GroupBy

```
AxB |> group_by(name.x) |> summarise(n=n())
```

Li	2
Wang	2

|> 它把右侧函数的第一个参数写到左侧。所以上面命令的等价于

```
summarise(group_by(AxB, name.x),n=n())
```

使用 |> 的语句更容易阅读，其表达力类似于 bash 中的管道（pipe，"|"）。

```
quote(AxB |> group_by(name.x) |> summarise(n=n()))
```

```
summarise(group_by(AxB, name.x), n = n())
```

GroupBy 两列

```
AxB |> group_by(name.x, ID.x) |> summarise(n=n())
```

Li	2	2
Wang	1	2

SortBy

```
order(AxB$name.x)
```

2
4
1
3

```
AxB[order(AxB$name.x),]
```

```
  ID.x name.x ID.y name.y
2    2     Li    2     Li
4    2     Li    3  Zhang
1    1   Wang    2     Li
3    1   Wang    3  Zhang
```

R 版本的统计

```
students <- read.csv("dataframe-practice/students.csv")
head(students)
```

```
  学号 姓名 性别   班级     联系方式
1    1   AB   女 物理71 50626922811
2    2   AC   女 工物61  6533879773
3    3   AD   男 物理71 43865400582
4    4   AE   男 工物71 58581462691
5    5   AF   女 物理72 45017508911
6    6   AG   男 工物72 26000243873
```

```
classes <- read.csv("dataframe-practice/classes.csv")
head(classes)
```

```
    班级 院系
1   核71 工物
2 工物61 工物
3 工物72 工物
4 基科71 物理
5 物理72 物理
6 物理71 物理
```

统计男女比例

```
merge(students, classes) |>
group_by(性别, 院系) |>
summarise(人数=n())
```

女	工物	4
女	物理	10
男	工物	18
男	物理	25

图形语法 Grammar of Graphics

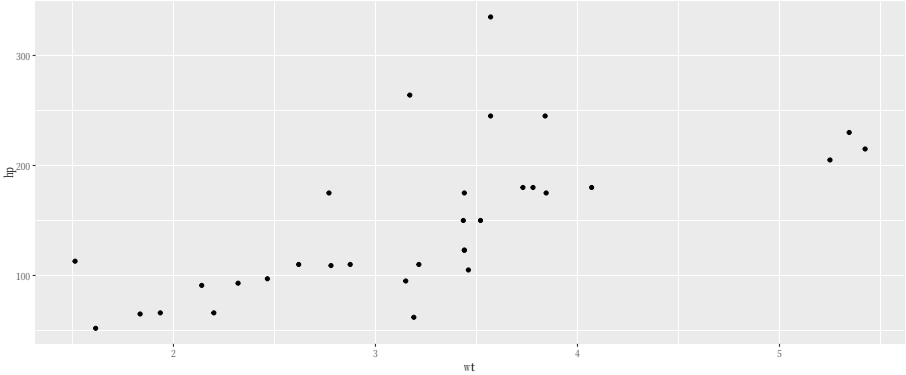
- 在表格与图形要素之间建立映射关系
- 表格对称的同等层次的列，都可以映射到各类图形要素中
  1. x 轴坐标
  2. y 轴坐标
  3. 颜色
  4. 点形状（圈、三角、方块等）
  5. 线形状（实线、虚线、点划线等）
  6. 点的大小
  7. 线的粗细
  8. 透明度
  9. 子图位置
- 在绘图时，通过调整映射关系来找到最佳的展现方式

参考书

- Leland Wilkinson, The Grammar of Graphics（沉痛缅怀 Wilkinson 教授）
- Hadley Wickham, A Layered Grammar of Graphics

ggplot

```
library(ggplot2)
p <- ggplot(mtcars, aes(x=wt, y=hp)) + geom_point()
print(p)
```



上色

```
p <- p + aes(color=am)
print(p)
```

ggplot 枚举型制图

```
mtcars$am <- factor(mtcars$am, levels=c(0, 1), labels=c("automatic", "manual"))
p <- p %+% mtcars
pdf("mtcars.pdf")
print(p)
dev.off()
```

表格的等价变换

- R 和 Pandas DataFrame 可以将长表与宽表相互转换
  - SQLite 不具备此功能
- melt 函数

使用 tidyr

- 发展历史为 reshape, reshape2, tidyr 。tidyr 是最新的库。

pivot_longer

- names_to 指定名字列，用于以枚举形式存储原有的列名。
- values_to 指定值例，用于存储原表的值。

```
library(tidyr)
scores <- read.csv("dataframe-practice/scores.csv")
```

```
scores |> pivot_longer(!学号, names_to="作业", values_to="分数") -> longscores
head(longscores)
```

1	self.intro	100
1	a.b	100
1	rank.guesser	100
1	hdf5	100
1	prime	100
1	heart.curve	120

变成枚举型

```
longscores$作业 <- as.factor(longscores$作业)
levels(longscores)
```

self.intro
a.b
rank.guesser
hdf5
prime
heart.curve
gpa.calculator
curve.fitting
大作业

加上大作业的类型

```
longscores$作业 <- longscores$作业 == "大作业"
summary(longscores$作业)
```

```
   Mode   FALSE    TRUE
logical     456      57
```

统计班级平均分

```
merge(students, longscores) |>
  group_by(班级, 作业) |>
  summarise(平均分=mean(分数)) |>
  ggplot(aes(x=作业, y=平均分, color=班级)) +
  geom_point() + facet_wrap(~班级)
```

could not find function "ggplot"

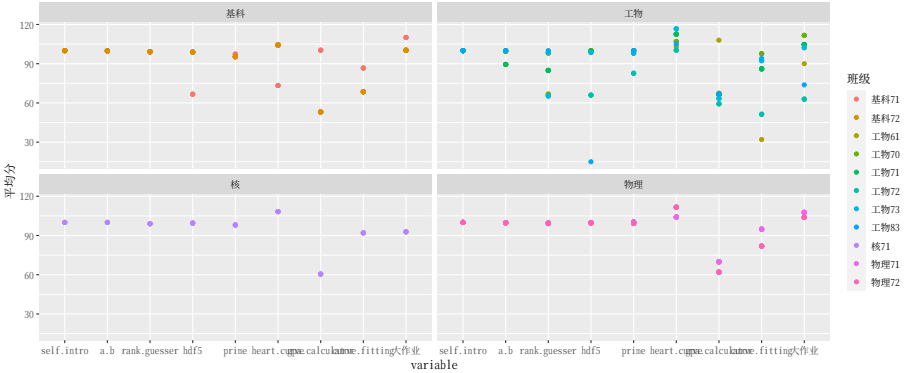
四个子图

```
students$班类 <- as.factor(gsub('[[:digit:]]*', '', students$班级))
unique(students$班类)
```

物理
工物
核
基科

统计班级平均分

```
merge(students, longscores) |>
  group_by(班级, 作业) |>
  reframe(平均分=mean(分数), 班类=班类) |>
  ggplot(aes(x=作业, y=平均分, color=班级)) +
  geom_point() + facet_wrap(~班类)
```



[100%] 少年爱迪生想

- ☒ 比较物理系和工物系的男女比例
- ☒ 算出大家的总评成绩：
  - ☒ 小作业权相等，总体占 65% 的成绩
  - ☒ 大作业占 30% 的成绩
  - ☒ 划分出不同的分数段，给出某分数段同学的手机号
- ☒ 画出各班平均小作业成绩的变化曲线

SQL 的 DataFrame 接口

```
library(dplyr)
library(dbplyr)
con <- DBI::dbConnect(RSQLite::SQLite(), dbname = "dataframe-practice/people.db")
classes <- tbl(con, "classes")
```

```
classes |> filter(院系=='物理')
```

基科 71	物理
物理 72	物理
物理 71	物理
基科 72	物理

查看 SQL 语句

```
sql_render(classes |> filter(院系=='物理'))
```

```
<SQL> SELECT *
FROM `classes`
WHERE (`院系` = '物理')
```