玩转数据处理120题 | Pandas&R

原创 刘早起、陈熹 早起Python 今天

本文精心挑选在数据处理中常见的120种操作并整理成习题发布。并且**每一题同时给出Pandas与R语言解法，同时针对部分习题给出了多种方法与注解**。本系列一共涵盖了数据处理、计算、可视化等常用操作，动手敲一遍代码一定会让你有所收获！

1 创建DataFrame

题目：将下面的字典创建为DataFrame

```
data = {"grammer":["Python","C","Java","GO",np.nan,"SQL","PHP","Python"],
        "score":[1,2,np.nan,4,5,6,7,10]}
```

难度：☆

期望结果

	grammer	score
0	Python	1.0
1	C	2.0
2	Java	NaN
3	GO	4.0
4	R	5.0
5	SQL	6.0
6	PHP	7.0
7	Python	10.0

Python解法

```
import numpy as np
import pandas as pd
df = pd.DataFrame(data)
```

```
# 假如是直接创建
df = pd.DataFrame({
    "grammer": ["Python","C","Java","GO",np.nan,"SQL","PHP","Python"],
    "score": [1,2,np.nan,4,5,6,7,10]})
```

R语言解法

```
# R中没有字典概念，故直接创建dataframe/tibble
#> 第一种
df <- data.frame(
  "grammer" = c("Python","C","Java","GO",NA,"SQL","PHP","Python"),
  "score" = c(1,2,NA,4,5,6,7,10)
)
#> 第二种
library(tibble)
df <- tibble(
  "grammer" = c("Python","C","Java","GO",NA,"SQL","PHP","Python"),
  "score" = c(1,2,NA,4,5,6,7,10)
)
# 也可以用tribble横向建tibble
```

注：1-20题均基于该数据框给出

2 数据提取

题目：提取含有字符串"Python"的行

难度：☆☆

期望结果

```
   grammer  score
0  Python     1.0
7  Python    10.0
```

Python解法：

```
#> 1
df[df['grammer'] == 'Python']
#> 2
results = df['grammer'].str.contains("Python")
results.fillna(value=False,inplace = True)
df[results]
```

R语言解法

```
df[which(df$grammer == 'Python'),]
```

3 提取列名

题目： 输出df的所有列名

难度： ☆

期望结果

```
Index(['grammer', 'score'], dtype='object')
```

Python解法

```
df.columns
```

R语言解法

```
names(df)
# [1] "grammer" "score"
```

4 修改列名

题目： 修改第二列列名为'popularity'

难度： ☆☆

Python解法

```
df.rename(columns={'score':'popularity'}, inplace = True)
```

R语言解法

```
df <- df %>%
  rename(popularity = score)
```

5 字符统计

题目： 统计grammer列中每种编程语言出现的次数

难度： ☆☆

Python解法

```
df['grammer'].value_counts()
```

R语言解法

```
# 神方法table
table(df$grammer)
```

6 缺失值处理

题目：将空值用上下值的平均值填充

难度：☆☆☆

Python解法

```
# pandas里有一个插值方法，就是计算缺失值上下两数的均值
df['popularity'] = df['popularity'].fillna(df['popularity'].interpolate(
```

R语言解法

```
library(Hmisc)
index <- which(is.na(df$popularity))
df$popularity <- impute(df$popularity,
                        (unlist(df[index-1, 2] +
                                df[index+1, 2]))/2)
```


7 数据提取

题目：提取popularity列中值大于3的行

难度：☆☆

Python解法

```
df[df['popularity'] > 3]
```

R语言解法

```
df %>%
  filter(popularity > 3)
# 等价于
df[df$popularity > 3,] # 这种方法跟pandas很相似
```

8 数据去重

题目：按照grammer列进行去重

难度：☆☆

Python解法

```
df.drop_duplicates(['grammer'])
```

R语言解法

```
df[!duplicated(df$grammer),]
```

9 数据计算

题目：计算popularity列平均值

难度：☆☆

Python解法

```
df['popularity'].mean()
# 4.75
```

R语言解法

```
#> 第一种
mean(df$popularity)
# [1] 4.75

#> 第二种
df %>%
  summarise(mean = mean(popularity))
## A tibble: 1 x 1
# mean
# <dbl>
# 1 4.75
```


10 格式转换

题目：将grammer列转换为list

难度：☆☆

Python解法

```
df['grammer'].to_list()
```

```
# ['Python', 'C', 'Java', 'GO', nan, 'SQL', 'PHP', 'Python']
```

R解法

```
unlist(df$grammer)
# [1] "Python" "C" "Java" "GO" NA "SQL" "PHP" "Python"
```

11 数据保存

题目：将DataFrame保存为EXCEL

难度：☆☆

Python解法

```
df.to_excel('filename.xlsx')
```

R解法

```
#R对EXCEL文件不太友好
#第一种方法：利用readr包转为csv再用EXCEL打开
#文件本质依然是csv
library(readr)
write_excel_csv(df,'filename.csv')

#第二种方法：利用openxlsx包
openxlsx::write.xlsx(df,'filename.xlsx')

#也可以用xlsx包，但需要先配置JAVA环境
#确保JAVA配置到环境变量中并命名为JAVA_HOME
Sys.getenv("JAVA_HOME")
install.packages('rJava')
install.packages("xlsxjars")
library(rJava)
library(xlsxjars)
xlsx::write.xlsx(df,'filename.xlsx')
```


12 数据查看

题目：查看数据行列数

难度：☆

Python解法

```
df.shape
# (8, 2)
```

R解法

```
dim(df)
# [1] 8 2
```

13 数据提取

题目：提取popularity列值大于3小于7的行

难度：☆☆

Python解法

```
df[(df['popularity'] > 3) & (df['popularity'] < 7)]
```

R解法

```
library(dplyr)
df %>%
  filter(popularity > 3 & popularity <7)
# 等价于
df[(df$popularity > 3) & (df$popularity <7),]
```


14 位置处理

题目：交换两列位置

难度：☆☆☆

Python解法

```
temp = df['popularity']
df.drop(labels=['popularity'], axis=1,inplace = True)
df.insert(0, 'popularity', temp)
```

R解法

```
df <- df %>%
    select(popularity,everything())
```

15 数据提取

题目：提取popularity列最大值所在行

难度：☆☆

Python解法

```
df[df['popularity'] == df['popularity'].max()]
```

R解法

```
df %>%
  filter(popularity == max(popularity))
# 同理也有类似pandas的方法
df[df$popularity == max(df$popularity),]
```

16 数据查看

题目：查看最后5行数据

难度：☆

Python解法

```
df.tail()
```

R解法

```
# R中head和tail默认是6行，可以指定数字
tail(df,5)
```


17 数据修改

题目：删除最后一行数据

难度：☆

Python解法

```
df = df.drop(labels=df.shape[0]-1)
```

R解法

```
df[-dim(df)[1],]
# 等价于
```

```
df %>%
  filter(rownames(df) != max(rownames(df)))
```

18 数据修改

题目：添加一行数据['Perl',6.6]

难度：☆☆

Python解法

```
row = {'grammer':'Perl','popularity':6.6}
df = df.append(row,ignore_index=True)
```

R解法

```
row <- c(6.6,'Perl') # 需要和列的位置对应
# 或者建数据框
row <- data.frame(
  "grammer" = c("Perl"),
  "popularity" = c(6.6)
)

df <- rbind(df,row)
```


19 数据整理

题目：对数据按照"popularity"列值的大小进行排序

难度：☆☆

Python解法

```
df.sort_values("popularity",inplace=True)
```

R解法

```
df <- df %>%
  arrange(popularity)
```

20 字符统计

题目：统计grammer列每个字符串的长度

难度：☆☆☆

Python解法

```
df['grammer'] = df['grammer'].fillna('R')
df['len_str'] = df['grammer'].map(lambda x: len(x))
```

R解法

```
library(Hmisc)
library(stringr)
df$grammer <- impute(df$grammer,'R')
str_length(df$grammer)

df$len_str <- str_length(df$grammer)
```

第二期：数据处理基础

21 数据读取

题目：读取本地EXCEL数据

难度：☆

Python解法

```
import pandas as pd
import numpy as np
df = pd.read_excel(r'C:\Users\chenx\Documents\Data Analysis\pandas120.xl
```

R解法

```
#R语言处理excel不友好，直接读取日期时间数据会变成实数
#openxlsx::read.xlsx中的detectDates参数只能识别纯日期
#as.Data转换该列后时间数据丢失，只有日期
#故先把excel文件转存为csv后用readr包读取
# 该方法不理想
library(openxlsx)
df <- read.xlsx('pandas120.xlsx',detectDates = T)
df$createTime <- as.Date(df$createTime,origin="1900-01-01")
# 转存csv后再读
library(readr)
df <- read_csv('pandas120.csv')
```

21—50部分习题与该数据相关

22 数据查看

题目：查看df数据前5行

难度：☆

期望输出

	createTime	education	salary
0	2020-03-16 11:30:18	本科	20k-35k
1	2020-03-16 10:58:48	本科	20k-40k
2	2020-03-16 10:46:39	不限	20k-35k
3	2020-03-16 10:45:44	本科	13k-20k
4	2020-03-16 10:20:41	本科	10k-20k

Python解法

```
df.head()
```

R解法

```
# 默认是6行，可指定行数
head(df,5)
```


23 数据计算

题目：将`salary`列数据转换为最大值与最小值的平均值

难度：☆☆☆☆

期望输出

	createTime	education	salary
0	2020-03-16 11:30:18	本科	27500
1	2020-03-16 10:58:48	本科	30000
2	2020-03-16 10:46:39	不限	27500
3	2020-03-16 10:45:44	本科	16500
4	2020-03-16 10:20:41	本科	15000

Python解法

```
# 方法一：apply + 自定义函数
def func(df):
    lst = df['salary'].split('-')
    smin = int(lst[0].strip('k'))
    smax = int(lst[1].strip('k'))
    df['salary'] = int((smin + smax) / 2 * 1000)
    return df

df = df.apply(func,axis=1)
# 方法二：iterrows + 正则
import re
for index,row in df.iterrows():
    nums = re.findall('\d+',row[2])
    df.iloc[index,2] = int(eval(f'({nums[0]} + {nums[1]}) / 2 * 1000'))
```

R解法

```
library(stringr)
df$salary <- df$salary %>%
  str_replace_all('k','') %>%
  str_split('-',simplify = T) %>%
  apply(2,as.numeric) %>%
  rowMeans() * 1000
```


24 数据分组

题目：将数据根据学历进行分组并计算平均薪资

难度：☆☆☆

期望输出

```
education salary
不限 19600.000000
大专 10000.000000
本科 19361.344538
硕士 20642.857143
```

Python解法

```
df.groupby('education').mean()
```

R解法

```
df %>%
  group_by(education) %>%
  summarise(mean = mean(salary))
```

25 时间转换

题目： 将createTime列时间转换为月-日

难度： ☆☆☆

期望输出

	createTime	education	salary
0	03-16	本科	27500
1	03-16	本科	30000
2	03-16	不限	27500
3	03-16	本科	16500
4	03-16	本科	15000

Python解法

```
for index,row in df.iterrows():
    df.iloc[index,0] = df.iloc[index,0].to_pydatetime().strftime("%m-%d")
```

R解法

```
#转化后该列属性是 字符串，R中对时间格式要求严格
df$createTime <- as.Date(df$createTime) %>%
  str_replace('2020-','')
```

26 数据查看

题目：查看索引、数据类型和内存信息

难度：☆

期望输出

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 135 entries, 0 to 134
Data columns (total 4 columns):
createTime 135 non-null object
education 135 non-null object
salary 135 non-null int64
categories 135 non-null category
dtypes: category(1), int64(1), object(2)
memory usage: 3.5+ KB
```

Python解法

```
df.info()
```

R解法

```
str(df)

# 内存查看需要用到其他的库
library(pryr)
object_size(df)
# 6.66 kB
```


27 数据查看

题目：查看数值型列的汇总统计

难度：☆

Python解法

```
df.describe()
```

R解法

```
summary(df)
```

28 数据整理

题目：新增一列根据salary将数据分为三组

难度：☆☆☆☆

输入

期望输出

	createTime	education	salary	categories
0	03-16	本科	27500	高
1	03-16	本科	30000	高
2	03-16	不限	27500	高
3	03-16	本科	16500	中
4	03-16	本科	15000	中

Python解法

```
bins = [0,5000, 20000, 50000]
group_names = ['低', '中', '高']
df['categories'] = pd.cut(df['salary'], bins, labels=group_names)
```

R解法

```
#用ifelse也可以
#底层原理有差别但实现结果一样
df <- df %>%
  mutate(categories = case_when(
    salary >= 0 & salary < 5000 ~ '低',
    salary >= 5000 & salary < 20000 ~ '低',
    TRUE ~ '高'
  ))
```


29 数据整理

题目：按照salary列对数据降序排列

难度：☆☆

Python解法

```
df.sort_values('salary', ascending=False)
```

R解法

```
df %>%
  arrange(desc(salary))
```

30 数据提取

题目：取出第33行数据

难度：☆☆

Python解法

```
df.iloc[32]
```

R解法

```
df[33,]
```

31 数据计算

题目：计算salary列的中位数

难度：☆☆

Python解法

```
np.median(df['salary'])
# 17500.0
```

R解法

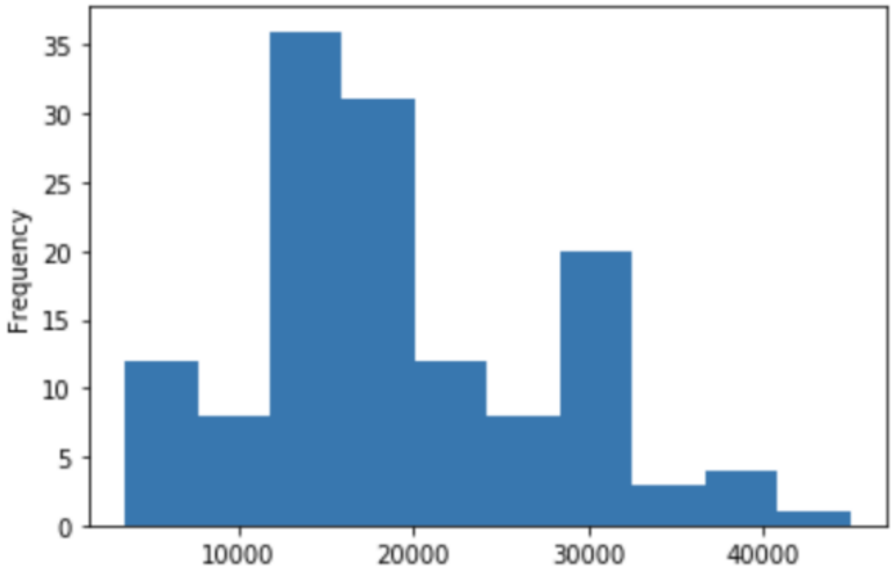
```
median(df$salary)
# [1] 17500
```

32 数据可视化

题目：绘制薪资水平频率分布直方图

难度：☆☆☆

期望输出



Python解法

```
# Jupyter运行matplotlib成像需要运行魔术命令
%matplotlib inline
plt.rcParams['font.sans-serif'] = ['SimHei'] # 解决中文乱码
plt.rcParams['axes.unicode_minus'] = False # 解决符号问题
import matplotlib.pyplot as plt
plt.hist(df.salary)

# 也可以用原生pandas方法绘图
df.salary.plot(kind='hist')
```

R解法

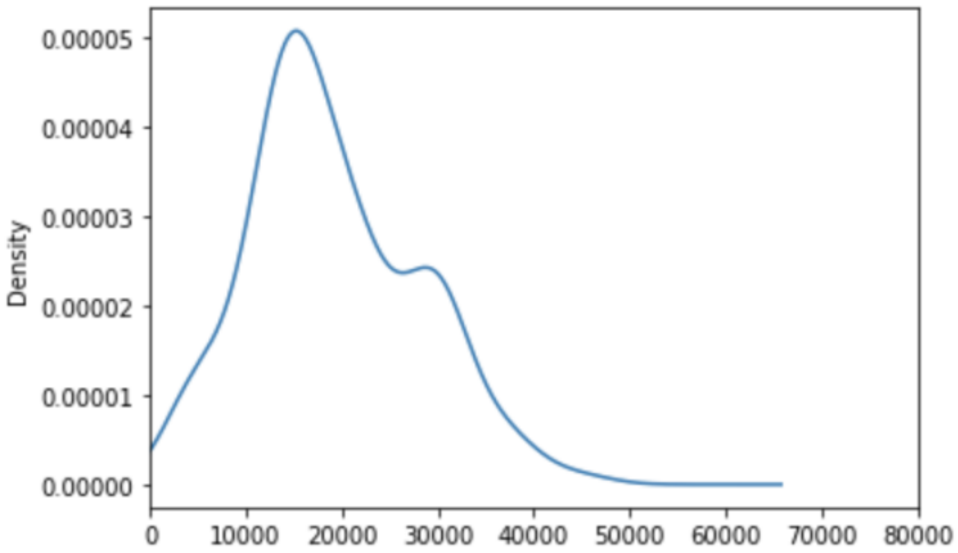
```
library(ggplot2)
library(patchwork)

df %>%
  ggplot(aes(salary)) +
  geom_histogram() +
  df %>%
  ggplot(aes(salary)) +
  geom_histogram(bins = 10) # 这个跟python的bins一致
```


33 数据可视化

题目：绘制薪资水平密度曲线

难度：☆☆☆

期望输出



Python解法

```
df.salary.plot(kind='kde',xlim = (0,70000))
```

R解法

```
df %>%
  ggplot(aes(salary)) +
  geom_density() +
  xlim(c(0,70000))
```


34 数据删除

题目：删除最后一列categories

难度：☆

Python解法

```
del df['categories']
# 等价于
df.drop(columns=['categories'], inplace=True)
```

R解法

```
df <- df[,-4]
# 提高可读性可采用如下代码
df <- df %>%
  select(-c('categories'))
```

35 数据处理

题目：将df的第一列与第二列合并为新的一列

难度：☆☆

Python解法

```
df['test'] = df['education'] + df['createTime']
```

R解法

```
df <- df %>%
  mutate(test = paste0(df$education,df$createTime))
```

36 数据处理

题目：将education列与salary列合并为新的一列

难度：☆☆☆

备注：salary为int类型，操作与35题有所不同

Python解法

```
df["test1"] = df["salary"].map(str) + df['education']
```

R解法

```
df <- df %>%
  mutate(test1 =
           paste0(df$salary,df$education))
```

37 数据计算

题目：计算salary最大值与最小值之差

难度：☆☆☆

Python解法

```
df[['salary']].apply(lambda x: x.max() - x.min())
# salary 41500
```

```
# dtype: int64
```

R解法

```
df %>%
  summarise(delta = max(salary) - min(salary)) %>%
  unlist()
# delta
# 41500
```

38 数据处理

题目：将第一行与最后一行拼接

难度：☆☆

Python解法

```
pd.concat([df[1:2], df[-1:]])
```

R解法

```
rbind(df[1,],df[dim(df)[1],])
```

39 数据处理

题目：将第8行数据添加至末尾

难度：☆☆

Python解法

```
df.append(df.iloc[7])
```

R解法

```
rbind(df,df[8,])
```

40 数据查看

题目：查看每列的数据类型

难度：☆

期望结果

```
createTime object
education object
salary int64
test object
test1 object
dtype: object
```

Python解法

```
df.dtypes
# createTime object
# education object
# salary int64
# test object
# test1 object
# dtype: object
```

R解法

```
str(df)
# tibble [135 x 5] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
# $ createTime: chr [1:135] "03-16" "03-16" "03-16" "03-16" ...
# $ education : chr [1:135] "本科" "本科" "不限" "本科" ...
# $ salary : num [1:135] 27500 30000 27500 16500 15000 14000 23000 12500
# $ test : chr [1:135] "本科03-16" "本科03-16" "不限03-16" "本科03-16" ...
# $ test1 : chr [1:135] "27500本科" "30000本科" "27500不限" "16500本科" ...
```


41 数据处理

题目：将createTime列设置为索引

难度：☆☆

Python解法

```
df.set_index("createTime")
```

R解法

```
df %>%
  tibble::column_to_rownames('createTime')
```

42 数据创建

题目：生成一个和df长度相同的随机数dataframe

难度：☆☆

Python解法

```
df1 = pd.DataFrame(pd.Series(np.random.randint(1, 10, 135)))
```

R解法

```
df1 <- sapply(135,function(n) {
  replicate(n,sample(1:10,1))
})
# 列名暂时不一样，下一题重命名
```

43 数据处理

题目：将上一题生成的dataframe与df合并

难度：☆☆

Python解法

```
df= pd.concat([df,df1],axis=1)
```

R解法

```
df <- cbind(df,df1) %>%
  rename(`0` = df1)
# 非常规命名需要用``包裹变量名
```


44 数据计算

题目：生成新的一列new为salary列减去之前生成随机数列

难度：☆☆

Python解法

```
df["new"] = df["salary"] - df[0]
```

R解法

```
df <- df %>%
  mutate(new = salary - `0`)
```

45 缺失值处理

题目：检查数据中是否含有任何缺失值

难度：☆☆☆

Python解法

```
df.isnull().values.any()
# False
```

R解法

```
# 这个包的结果呈现非常有趣
library(mice)
md.pattern(df)
```

46 数据转换

题目：将salary列类型转换为浮点数

难度：☆☆☆

Python解法

```
df['salary'].astype(np.float64)
```

R解法

```
as.double(df2$salary)
```


47 数据计算

题目：计算salary大于10000的次数

难度：☆☆

Python解法

```
len(df[df['salary'] > 10000])
# 119
```

R解法

```
df %>%
  filter(salary > 10000) %>%
```

```
dim(.) %>%
.[1]
```

48 数据统计

题目：查看每种学历出现的次数

难度：☆☆☆

期望输出

```
本科 119
硕士 7
不限 5
大专 4
Name: education, dtype: int64
```

Python解法

```
df.education.value_counts()
```

R解法

```
table(df$education)
```


49 数据查看

题目：查看education列共有几种学历

难度：☆☆

Python解法

```
df['education'].nunique()
# 4
```

R解法

```
length(unique(df$education))
# [1] 4
```

50 数据提取

题目：提取salary与new列的和大于60000的最后3行

难度：☆☆☆☆

期望输出

	createTime	education	salary	test	test1	0	new
92	03-16	本科	35000	本科03-16	35000本科	6	34994
101	03-16	本科	37500	本科03-16	37500本科	5	37495
131	03-16	硕士	37500	硕士03-16	37500硕士	6	37494

Python解法

```
rowsums = df[['salary','new']].apply(np.sum, axis=1)
res = df.iloc[np.where(rowsums > 60000)[0][-3:], :]
```

R解法

```
df[df$salary + df$new > 60000,] %>%
  .[nrow(.)-3+1:nrow(.),] %>%
  na.omit(.)
```

51 数据读取

题目：使用绝对路径读取本地Excel数据

难度：☆

Python解法

```
import pandas as pd
import numpy as np

df = pd.read_excel(r'C:\Users\chenx\Documents\Data Analysis\Pandas51-80.
```

R解法

```
# 转存csv后再读
library(readr)
df <- read_csv('C:/Users/chenx/Documents/Data Analysis/Pandas51-80.csv')
```

备注

请将答案中路径替换为自己机器存储数据的绝对路径，51—80相关习题与该数据有关

52 数据查看

题目：查看数据前三行

难度：☆

期望结果

	代码	简称	日期	前收盘价(元)	开盘价(元)	最高价(元)	最低价(元)	收盘价(元)	成交量(股)	成交金额(元)	涨跌(元)	涨跌幅(%)	均价(元)	换手率(%)	A股流通市值(元)	总市值(元)
0	600000.SH	浦发银行	2016-01-04	16.1356	16.1444	16.1444	15.4997	15.7205	42240610	754425783	-0.4151	-2.5725	17.8602	0.2264	3.320318e+11	3.320318e+11
1	600000.SH	浦发银行	2016-01-05	15.7205	15.4644	15.9501	15.3672	15.8618	58054793	1034181474	0.1413	0.8989	17.8139	0.3112	3.350163e+11	3.350163e+11
2	600000.SH	浦发银行	2016-01-06	15.8618	15.8088	16.0208	15.6234	15.9855	46772653	838667398	0.1236	0.7795	17.9307	0.2507	3.376278e+11	3.376278e+11

Python解法

```
df.head(3)
```

R解法

```
head(df,3)
```


53 缺失值处理

题目：查看每列数据缺失值情况

难度：☆☆

期望结果

代码 **1**
简称 **2**
日期 **2**
前收盘价(元) **2**
开盘价(元) **2**
最高价(元) **2**
最低价(元) **2**
收盘价(元) **2**
成交量(股) **2**
成交金额(元) **2**
.....

Python解法

```
df.isnull().sum()
```

R解法

```
colSums(is.na(df))
```

54 缺失值处理

题目：提取日期列含有空值的行

难度：☆☆

期望结果

	代码	简称	日期	前收盘价(元)	开盘价(元)	最高价(元)	最低价(元)	收盘价(元)	成交量(股)	成交金额(元)	涨跌(元)	涨跌幅(%)	均价(元)	换手率(%)	A股流通市值(元)	总市值(元)	A股流通股本(股)	市盈率
327	NaN	NaN	NaT	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
328	数据来源：Wind资讯	NaN	NaT	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Python解法

```
df[df['日期'].isnull()]
```

R解法

```
df[is.na(df$日期),]
```


55 缺失值处理

题目：输出每列缺失值具体行数

难度：☆☆☆

期望结果

```
列名："代码"，第[327]行位置有缺失值
列名："简称"，第[327，328]行位置有缺失值
列名："日期"，第[327，328]行位置有缺失值
列名："前收盘价(元)"，第[327，328]行位置有缺失值
列名："开盘价(元)"，第[327，328]行位置有缺失值
列名："最高价(元)"，第[327，328]行位置有缺失值
列名："最低价(元)"，第[327，328]行位置有缺失值
列名："收盘价(元)"，第[327，328]行位置有缺失值
.................
```

Python解法

```
for i in df.columns:
    if df[i].count() != len(df):
        row = df[i][df[i].isnull().values].index.tolist()
        print('列名："{}"，第{}行位置有缺失值'.format(i,row))
```

R解法

```
library(glue)

for (i in names(df)){
  if(sum(is.na(df[,'日期'])) != 0){
    res1 <- which(is.na(df[,i]))
    res2 <- paste(res1,collapse = ',')
    print(glue('列名："{i}"，第[{res2}]行有缺失值'))
  }
}
```

56 缺失值处理

题目：删除所有存在缺失值的行

难度：☆☆

Python解法

```
df.dropna(axis=0, how='any', inplace=True)
```

R解法

```
df <- na.omit(df)
```

备注

axis：0-行操作（默认），1-列操作
how：any-只要有空值就删除（默认），all-全部为空值才删除
inplace：**False**-返回新的数据集（默认），**True**-在原数据集上操作

57 数据可视化

题目：绘制收盘价的折线图

难度：☆☆

期望结果

Python解法

```
# Jupyter运行matplotlib
%matplotlib inline

df['收盘价(元)'].plot()
# 等价于
import matplotlib.pyplot as plt
plt.plot(df['收盘价(元)'])
```

R解法

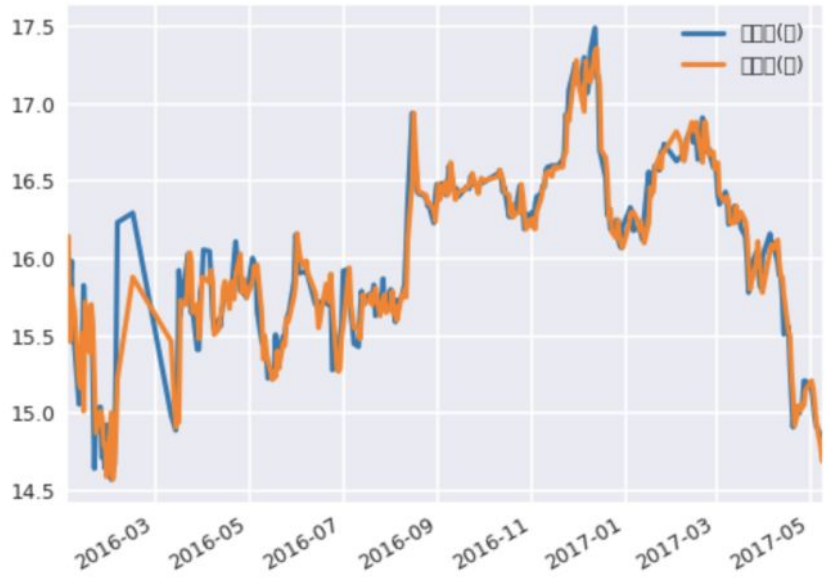
```
library(ggplot2)

df %>%
  ggplot(aes(日期,`收盘价(元)`)) +
  geom_line()
```


58 数据可视化

题目：同时绘制开盘价与收盘价

难度：☆☆☆

期望结果



Python解法

```
plt.rcParams['font.sans-serif'] = ['SimHei'] # 解决中文乱码
plt.rcParams['axes.unicode_minus'] = False # 解决符号问题

df[['收盘价(元)','开盘价(元)']].plot()
```

R解法

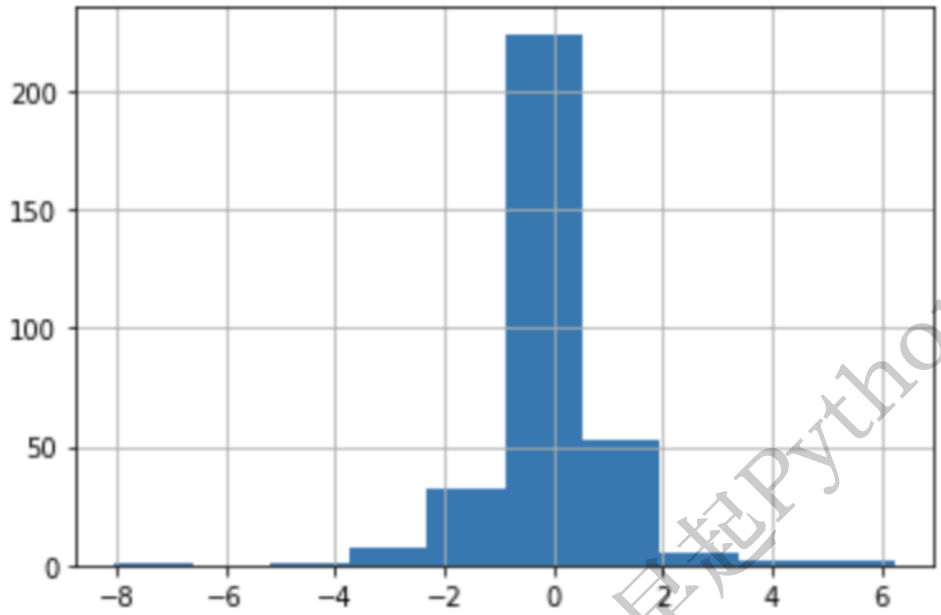
```
df %>%
  ggplot() +
  geom_line(aes(日期,`收盘价(元)`), size=1.2, color='steelblue') +
  geom_line(aes(日期,`开盘价(元)`), size=1.2, color='orange') +
  ylab(c('价格(元)'))
# 这种画出来没有图例，当然可以手动添加，但为了映射方便可以用另一种方法

library(tidyr)

df %>%
  select(日期,`开盘价(元)`,`收盘价(元)`) %>%
  pivot_longer(c(`开盘价(元)`,`收盘价(元)`),
               names_to='type',values_to='price') %>%
  ggplot(aes(日期,price,color=type)) +
  geom_line(size=1.2) +
  scale_color_manual(values=c('steelblue','orange')) +
  theme_bw() +
  theme(
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    legend.title = element_blank(),
    legend.position = c(0.86, 0.9)
  )
```

59 数据可视化

题目：绘制涨跌幅的直方图

难度：☆☆

期望结果



Python解法

```
plt.hist(df['涨跌幅(%)'])
# 等价于
df['涨跌幅(%)'].hist()
```

R解法

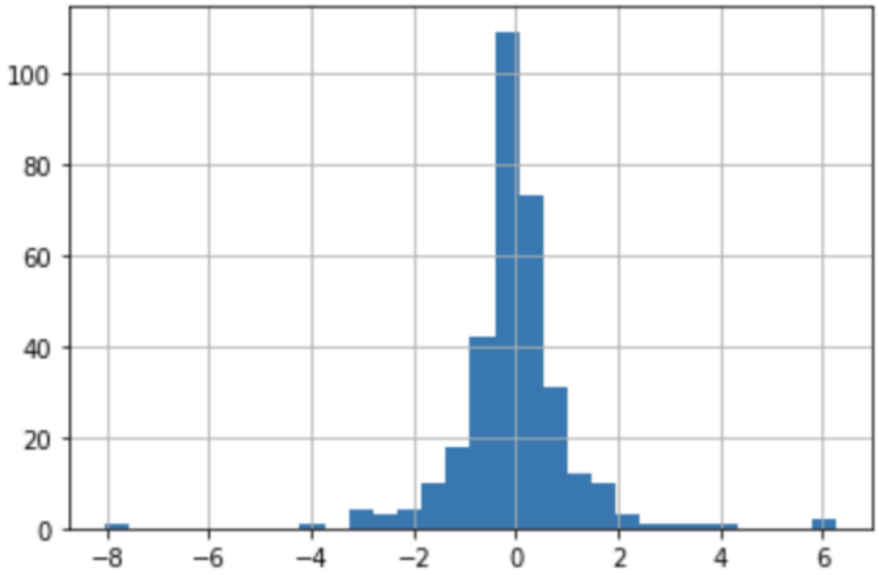
```
df %>%
  ggplot(aes(`涨跌幅(%)`)) +
  geom_histogram()
# 可以指定bins
```


60 数据可视化

题目：让直方图更细致

难度：☆☆

期望结果



Python解法

```
df['涨跌幅(%)'].hist(bins = 30)
```

R解法

```
df %>%
  ggplot(aes(`涨跌幅(%)`)) +
  geom_histogram(bins=30)
```

61 数据创建

题目： 以data的列名创建一个dataframe

难度： ☆☆

Python解法

```
temp = pd.DataFrame(columns = df.columns.to_list())
```

R解法

```
temp <- as_tibble(names(df))
```

62 异常值处理

题目： 打印所有换手率不是数字的行

难度： ☆☆☆

期望结果

	代码	简称	日期	前收盘价(元)	开盘价(元)	最高价(元)	最低价(元)	收盘价(元)	成交量(股)	成交金额(元)	涨跌(元)	涨跌幅(%)	均价(元)	换手率(%)	A股流通市值(元)	总市值(元)	A股流通股本(股)	市盈率
26	600000.SH	浦发银行	2016-02-16	16.2946	16.2946	16.2946	16.2946	16.2946	--	--	0.0	0.0	--	--	3.441565e+11	3.441565e+11	1.865347e+10	6.801
27	600000.SH	浦发银行	2016-02-17	16.2946	16.2946	16.2946	16.2946	16.2946	--	--	0.0	0.0	--	--	3.441565e+11	3.441565e+11	1.865347e+10	6.801
28	600000.SH	浦发银行	2016-02-18	16.2946	16.2946	16.2946	16.2946	16.2946	--	--	0.0	0.0	--	--	3.441565e+11	3.441565e+11	1.865347e+10	6.801

Python解法

```
for index,row in df.iterrows():
    if type(row[13]) != float:
        temp = temp.append(df.loc[index])
```

R解法

```
#换手率这一列属性为chr，需要先强转数值型
#如果转换失败会变成NA，判断即可
df[is.na(as.numeric(df$`换手率(%)`)),]
```

63 异常值处理

题目： 打印所有换手率为--的行

难度： ☆☆☆

Python解法

```
df[df['换手率(%)'] == '--']
```

R解法

```
df %>%
  filter(`换手率(%)` == '--')
```

备注

通过上一题我们发现换手率的异常值只有--

64 数据处理

题目： 重置data的行号

难度： ☆

Python解法

```
df = df.reset_index(drop=True)
```

R解法

```
rownames(df) <- NULL

# 如果是tibble则索引始终是按顺序
```

备注

有时我们修改数据会导致索引混乱

65 异常值处理

题目：删除所有换手率为非数字的行

难度：☆☆☆

Python解法

```
lst = []
for index,row in df.iterrows():
    if type(row[13]) != float:
        lst.append(index)
df.drop(labels=lst,inplace=True)
```

R解法

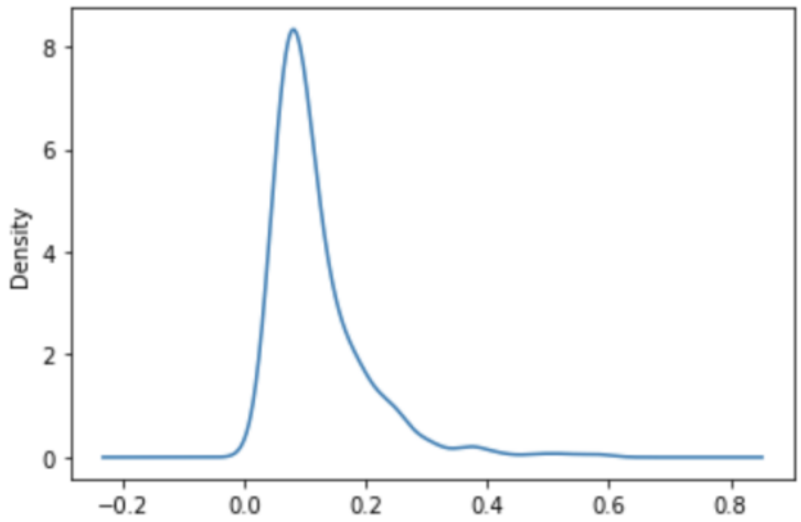
```
df[!is.na(as.numeric(df$`换手率(%)`)),]
# 或者根据前几题的经验，非数字就是'--'
df <- df %>%
  filter(`换手率(%)` != '--')
```


66 数据可视化

题目：绘制换手率的密度曲线

难度：☆☆☆

期望结果



Python解法

```
df['换手率(%)'].plot(kind='kde',xlim=(0,0.6))
```

R解法

```
df$`换手率(%)` <- as.double(df$`换手率(%)`)
ggplot(df) +
  geom_density(aes(`换手率(%)`))
```

67 数据计算

题目： 计算前一天与后一天收盘价的差值

难度： ☆☆

Python解法

```
df['收盘价(元)'].diff()
```

R解法

```
df %>%
  summarise(delta = `收盘价(元)` - lag(`收盘价(元)`))
```

68 数据计算

题目： 计算前一天与后一天收盘价变化率

难度： ☆☆

Python解法

```
data['收盘价(元)'].pct_change()
```

R解法

```
df %>%
  summarise(pct_change = (`收盘价(元)` - lag(`收盘价(元)`))/lag(`收盘价(元)`
```

69 数据处理

题目： 设置日期为索引

难度： ☆

Python解法

```
df.set_index('日期')
```

R解法

```
df %>%
  column_to_rownames(var='日期')
```

70 指标计算

题目： 以5个数据作为一个数据滑动窗口，在这个5个数据上取均值(收盘价)

难度： ☆☆☆

Python解法

```
df['收盘价(元)'].rolling(5).mean()
```

R解法

```
library(RcppRoll)
df %>%
  transmute(avg_5 = roll_mean(`收盘价(元)`,n = 5,align="right",fill = NA)
```

71 指标计算

题目： 以5个数据作为一个数据滑动窗口，计算这五个数据总和(收盘价)

难度： ☆☆☆

Python解法

```
df['收盘价(元)'].rolling(5).sum()
```

R解法

```
df %>%
  transmute(sum_5 = roll_sum(`收盘价(元)`,n = 5,align="right",fill = NA))
```

72 数据可视化

题目： 将收盘价5日均线、20日均线与原始数据绘制在同一个图上

难度： ☆☆☆

期望结果

Python解法

```
df['收盘价(元)'].plot()
df['收盘价(元)'].rolling(5).mean().plot()
df['收盘价(元)'].rolling(20).mean().plot()
```

R解法

```
df %>%
  mutate(avg_5 = roll_mean(`收盘价(元)`,n = 5,align="right",fill = NA),
         avg_20 = roll_mean(`收盘价(元)`,n = 20,align="right",fill = NA))
  ggplot() +
  geom_line(aes(日期,`收盘价(元)`),color = 'steelblue',size = 1.2) +
  geom_line(aes(日期,avg_5),color = 'orange',size = 1.2) +
  geom_line(aes(日期,avg_20),color = 'green',size = 1.2)
```

73 数据重采样

题目： 按周为采样规则，取一周收盘价最大值

难度： ☆☆☆

Python解法

```
df = df.set_index('日期')
df['收盘价(元)'].resample('W').max()
```

R解法

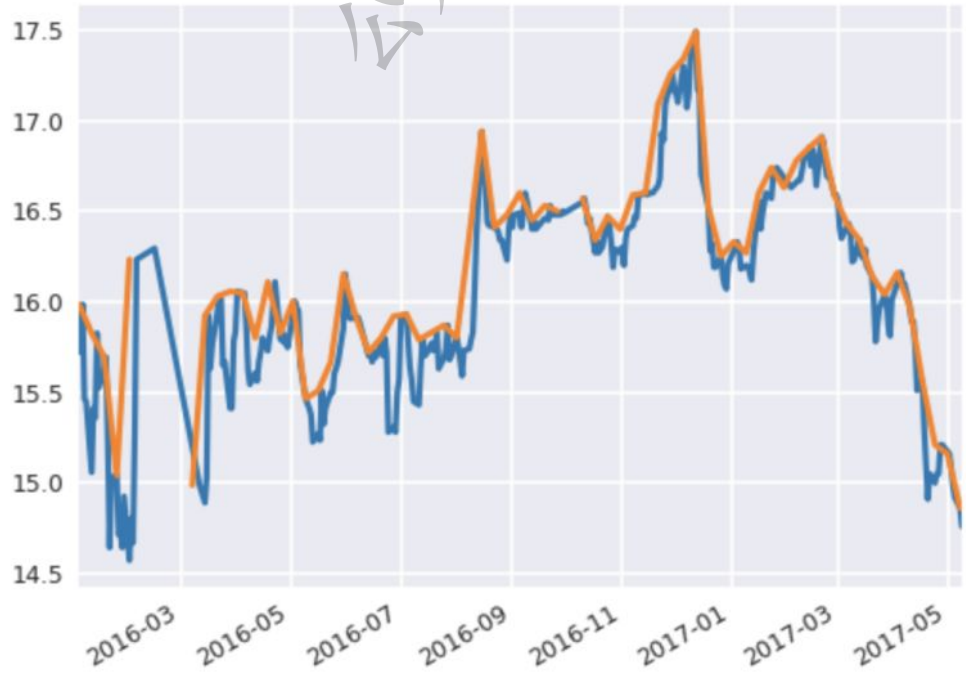
```
library(plyr)

res <- dlply(df,.(cut(日期,"1 week")),"[")
res_max <- sapply(res,function(n)max(n$`收盘价(元)`),simplify=TRUE)
as.data.frame(res_max)
```

74 数据可视化

题目： 绘制重采样数据与原始数据

难度： ☆☆☆

期望结果



Python解法

```
df['收盘价(元)'].plot()
df['收盘价(元)'].resample('7D').max().plot()
```

R解法

```
res %>%
  rownames_to_column('date')
res$date <- as.Date(res$date)

ggplot(df) +
  geom_line(aes(日期,`收盘价(元)`),color = 'steelblue',size = 1.2) +
  geom_line(data = res, aes(date,res_max),
            color = 'orange',size = 1.2)
```

75 数据处理

题目：将数据往后移动5天

难度：☆☆

Python解法

```
df.shift(5)
```

R解法

```
lag(df,5)
```


76 数据处理

题目：将数据向前移动5天

难度：☆☆

Python解法

```
df.shift(-5)
```

R解法

```
lead(df,5)
```

77 数据计算

题目：使用expending函数计算开盘价的移动窗口均值

难度：☆☆

Python解法

```
df['开盘价(元)'].expanding(min_periods=1).mean()
```

R解法

```
#R中没有expanding完全一致的函数
#考虑到expanding实际功能就是累积均值
#可以用cummean
#但cummean的功能和我预想的不同
#可能是包之间相互干扰
#最后采用cumsum/1:n的形式完成本题

res <- df %>%
  transmute(cummean = cumsum(`开盘价(元)`)/1:dim(df)[1])
```

78 数据可视化

题目：绘制上一题的移动均值与原始数据折线图

难度：☆☆☆

期望结果

Python解法

```
df['expanding Open mean']=df['开盘价(元)'].expanding(min_periods=1).mean(
df[['开盘价(元)', 'expanding Open mean']].plot(figsize=(16, 6))
```

R解法

```
library(tidyr)
df %>%
  cbind(res) %>%
  dplyr::rename(Opening_Price = `开盘价(元)`,
         Expanding_Open_Mean = cummean) %>%
  select(日期,Opening_Price,Expanding_Open_Mean) %>%
  pivot_longer(c(Opening_Price,Expanding_Open_Mean),
               names_to = 'type',
               values_to ='price') %>%
  ggplot(aes(日期,price,color = type)) +
  geom_line(size=1.2) +
  scale_color_manual(values=c('orange','steelblue')) +
  theme_bw() +
  theme(
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    legend.title = element_blank(),
    legend.position = c(0.9, 0.9)
  )
```

79 数据计算

题目：计算布林指标

难度：☆☆☆☆

Python解法

```
df['former 30 days rolling Close mean']=df['收盘价(元)'].rolling(20).mean
df['upper bound']=df['former 30 days rolling Close mean']+2*df['收盘价(元
df['lower bound']=df['former 30 days rolling Close mean']-2*df['收盘价(元
```

R解法

```
df <- df %>%
  mutate(avg_20 = roll_mean(`收盘价(元)`,n = 20,align="right",fill = NA),
         upper_bound = avg_20 + 2 * roll_sd(`收盘价(元)`,n = 20,align="ri
         lower_bound = avg_20 - 2 * roll_sd(`收盘价(元)`,n = 20,align="ri
```

80 数据可视化

题目：计算布林线并绘制

难度： ☆☆☆

期望结果

Python解法

```
df[['收盘价(元)', 'former 30 days rolling Close mean','upper bound','lowe
```

R解法

```
df %>%
  dplyr::rename(former_30_days_rolling_Close_mean = avg_20,
                Closing_Price = `收盘价(元)`) %>%
  select(日期,Closing_Price,
           former_30_days_rolling_Close_mean,upper_bound,lower_bound) %>
  pivot_longer(c(Closing_Price,former_30_days_rolling_Close_mean,upper_b
               names_to = 'type',
               values_to ='price') %>%
  ggplot(aes(日期,price,color = type)) +
  geom_line(size=1.2) +
  scale_color_manual(values=c('steelblue','orange','red','green')) +
  theme_bw() +
  theme(
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    legend.title = element_blank(),
    legend.position = c(0.6, 0.2)
  )
```

题目：导入并查看pandas与numpy版本

难度：☆

Python解法

```
import pandas as pd
import numpy as np
print(np.__version__)
# 1.16.5
print(pd.__version__)
# 0.25.1
```

R语言解法

```
packageVersion("tidyverse")
# [1] '1.3.0'
packageVersion("dplyr")
# [1] '0.8.99.9002'
```

82 数据创建

题目：从NumPy数组创建DataFrame

难度：☆

备注

使用numpy生成20个0-100随机数

Python解法

```
tem = np.random.randint(1,100,20)
df1 = pd.DataFrame(tem)
```

R语言解法

```
df1 <- sapply(20,function(n) {
  replicate(n,sample(1:100,1))
}) %>%
  as.data.frame(.) %>%
  dplyr::rename(`0` = V1)
```

83 数据创建

题目：从NumPy数组创建DataFrame

难度： ☆

备注

使用numpy生成20个0-100固定步长的数

Python解法

```
tem = np.arange(0,100,5)
df2 = pd.DataFrame(tem)
```

R语言解法

```
df2 <- as.data.frame(seq(0,99,5)) %>%
  dplyr::rename(`0` = "seq(0, 99, 5)")
```

84 数据创建

题目： 从NumPy数组创建DataFrame

难度： ☆

备注

使用numpy生成20个**指定分布(如标准正态分布)**的数

Python解法

```
tem = np.random.normal(0, 1, 20)
df3 = pd.DataFrame(tem)
```

R语言解法

```
df3 <- as.data.frame(rnorm(20,0,1)) %>%
  dplyr::rename(`0` = "rnorm(20, 0, 1)")
```


85 数据创建

题目： 将df1，df2，df3按照行合并为新DataFrame

难度： ☆☆

Python解法

```
df = pd.concat([df1,df2,df3],axis=0,ignore_index=True)
```

R语言解法

```
df <- rbind(df1,df2,df3)
```

86 数据创建

题目：将df1，df2，df3按照列合并为新DataFrame

难度：☆☆

期望结果

```
0 1 2
0 95 0 0.022492
1 22 5 -1.209494
2 3 10 0.876127
3 21 15 -0.162149
4 51 20 -0.815424
5 30 25 -0.303792
...............
```

Python解法

```
df = pd.concat([df1,df2,df3],axis=1,ignore_index=True)
```

R语言解法

```
df <- cbind(df1,df2,df3)
names(df) <- c(0,1,2)
```


87 数据查看

题目：查看df所有数据的最小值、25%分位数、中位数、75%分位数、最大值

难度：☆☆

Python解法

```
np.percentile(df, q=[0, 25, 50, 75, 100])
```

R语言解法

```
summary(unlist(df))
```

88 数据修改

题目：修改列名为col1,col2,col3

难度：☆

Python解法

```
df.columns = ['col1','col2','col3']
```

R语言解法

```
df <- df %>%
  dplyr::rename(col1 = 1,
                col2 = 2,
                col3 = 3)
# 或者用类似pandas的方法
names(df) <- c('col1','col2','col3')
```

89 数据提取

题目：提取第一列中不在第二列出现的数字

难度：☆☆☆

Python解法

```
df['col1'][~df['col1'].isin(df['col2'])]
```

R语言解法

```
df[!(df$col1 %in% df$col2),1]
```


90 数据提取

题目：提取第一列和第二列出现频率最高的三个数字

难度：☆☆☆

Python解法

```
temp = df['col1'].append(df['col2'])
temp.value_counts()[:3]
```

R语言解法

```
count(unlist(c(df$col1,df$col2))) %>%
  arrange(desc(freq)) %>%
  filter(row_number() <= 3)
```

91 数据提取

题目：提取第一列中可以整除5的数字位置

难度：☆☆☆

Python解法

```
np.argwhere(df['col1'] % 5==0)
```

R语言解法

```
which(df['col1'] %% 5==0)
```

92 数据计算

题目：计算第一列数字前一个与后一个的差值

难度：☆☆

Python解法

```
df['col1'].diff().tolist()
```

R语言解法

```
df %>%
  summarise(col1 - lag(col1)) %>%
  na.omit(.) # 不去NA也可以，pandas没有去除
```


93 数据处理

题目：将col1,col2,clo3三列顺序颠倒

难度：☆☆

Python解法

```
df.iloc[:, ::-1]
```

R语言解法

```
df %>%
  select(col3,col2,everything())
```

94 数据提取

题目：提取第一列位置在1,10,15的数字

难度：☆☆

Python解法

```
df['col1'].take([1,10,15])
# 等价于
df.iloc[[1,10,15],0]
```

R语言解法

```
df[c(1,10,15) + 1,1]
```

95 数据查找

题目：查找第一列的局部最大值位置

难度：☆☆☆☆

备注

即比它前一个与后一个数字的都大的数字

Python解法

```
res = np.diff(np.sign(np.diff(df['col1'])))
np.where(res== -2)[0] + 1
# array([ 2, 4, 7, 9, 12, 15], dtype=int64)
```

R语言解法

```
res1 <- which((df$col1 - lag(df$col1) > 0))
res2 <- which((df$col1 - lead(df$col1) > 0))

intersect(res1,res2)
# [1] 3 5 7 12 14 17 19

# 另一种方法，类似pandas的用符号判断

res <- sign(df$col1 - lag(df$col1))

which(res - lag(res) == -2) - 1
```

```
# # [1] 3 5 7 12 14 17 19
```

96 数据计算

题目：按行计算df的每一行均值

难度：☆☆

Python解法

```
df[['col1','col2','col3']].mean(axis=1)
```

R语言解法

```
rowMeans(df)
```

97 数据计算

题目：对第二列计算移动平均值

难度：☆☆☆

备注

每次移动三个位置，不可以使用自定义函数

Python解法

```
np.convolve(df['col2'], np.ones(3)/3, mode='valid')
```

R语言解法

```
library(RcppRoll)

df %>%
  summarise(avg_3 = roll_mean(col2, n=3))
```

98 数据修改

题目：将数据按照第三列值的大小升序排列

难度：☆☆

Python解法

```
df.sort_values("col3",inplace=True)
```

R语言解法

```
df <- df %>%
  arrange(col3)
```

99 数据修改

题目：将第一列大于50的数字修改为'高'

难度：☆☆

Python解法

```
df.col1[df['col1'] > 50] = '高'
```

R语言解法

```
df[df$col1 > 50,1] <- '高'
```

100 数据计算

题目：计算第一列与第二列之间的欧式距离

难度：☆☆☆

备注

不可以使用自定义函数

Python解法

```
np.linalg.norm(df['col1']-df['col2'])
# 194.29873905921264
```

R语言解法

```
# 可以利用概念计算
res <- (df$col1 - df$col2) ^ 2
sqrt(sum(res))
# [1] 197.0102

# 也可以利用dist函数，但需要形成两个不同的观测
dist(rbind(df$col1,df$col2))
# 1
# 2 197.0102
```

101 数据读取

题目：从CSV文件中读取指定数据

难度：☆☆

备注

从数据1中的前10行中读取positionName, salary两列

Python解法

```
df1 = pd.read_csv(r'C:\Users\chenx\Documents\Data Analysis\数据1.csv',en
```

R语言解法

```
#一步读取文件的指定列用readr包或者原生函数都没办法
#如果文件特别大又不想全部再选指定列可以用如下办法
#基本思想先读取较少的数据获取列名
#给目标列以外的列打上NULL导致第二次读取文件时NULL列丢失即可

res <- read.csv('数据1.csv',encoding = 'GBK',nrows = 3)
classes <- sapply(res, class)
classes[-match(c('positionName','salary'),names(classes))] <-
  rep('NULL', length(classes) - 2)

df <- read.csv('数据1.csv',encoding = 'GBK',nrows = 10,
               colClasses = classes)
```


102 数据读取

题目：从CSV文件中读取指定数据

难度：☆☆

备注

从数据2中读取数据并在读取数据时将薪资大于10000的为改为高

Python解法

```
df2 = pd.read_csv(r'C:\Users\chenx\Documents\Data Analysis\数据2.csv',
                  converters={'薪资水平': lambda x: '高' if float(x) > 10(
```

R语言解法

```
library(readr)
```

```
df2 <- read_csv('数据2.csv') %>%
  mutate('学历要求',
         '薪资水平' = ifelse(
           薪资水平 > 10000,'高','低'))
```

103 数据计算

题目：从dataframe提取数据

难度：☆☆☆

备注

从上一题数据中，对薪资水平列每隔20行进行一次抽样

期望结果

	薪资水平
0	高
20	高
40	高
60	高
80	高
100	高
120	高
140	高
160	高
180	高

Python解法

```
df2.iloc[::20, :][['薪资水平']]
```

R语言解法

```
df2[seq(1,dim(df2)[1],20),]
```

104 数据处理

题目：将数据取消使用科学计数法

难度： ☆☆

输入

```
df = pd.DataFrame(np.random.random(10)**10, columns=['data'])
```

期望结果

	data
0	0.078
1	0.029
2	0.002
3	0.000
4	0.000
5	0.000
6	0.007
7	0.000
8	0.000
9	0.004

Python解法

```
df = pd.DataFrame(np.random.random(10)**10, columns=['data'])
df.round(3)
```

R语言解法

```
df <- tibble(data = runif(10)^10)
round(df,3)
```


105 数据处理

题目： 将上一题的数据转换为百分数

难度： ☆☆☆

期望结果

	data
0	7.75%
1	2.94%
2	0.22%
3	0.00%
4	0.00%
5	0.00%
6	0.65%
7	0.01%
8	0.00%
9	0.38%

Python解法

```
df.style.format({'data': '{0:.2%}'.format})
```

R语言解法

```
tibble(data = str_glue('{round(df$data * 100,2)}%'))
```

106 数据查找

题目：查找上一题数据中第3大值的行号

难度：☆☆☆

Python解法

```
df['data'].argsort()[len(df)-3]
```

R语言解法

```
df %>%
  mutate(nrow = rownames(.)) %>%
  arrange(desc(data)) %>%
  filter(row_number() == 3) %>%
  select(nrow)
```

107 数据处理

题目：反转df的行

难度：☆☆

Python解法

```
df.iloc[::-1, :]
```

R语言解法

```
df %>%
  arrange(desc(rownames(.)))
```

108 数据重塑

题目：按照多列对数据进行合并

难度：☆☆

输入

```
df1= pd.DataFrame({'key1': ['K0', 'K0', 'K1', 'K2'],
'key2': ['K0', 'K1', 'K0', 'K1'],
'A': ['A0', 'A1', 'A2', 'A3'],
'B': ['B0', 'B1', 'B2', 'B3']})

df2= pd.DataFrame({'key1': ['K0', 'K1', 'K1', 'K2'],
'key2': ['K0', 'K0', 'K0', 'K0'],
'C': ['C0', 'C1', 'C2', 'C3'],
'D': ['D0', 'D1', 'D2', 'D3']})
```

Python解法

```
pd.merge(df1, df2, on=['key1', 'key2'])
```

R语言解法

```
df1 <- data.frame(
  "key1" = c("K0","K0","K1","K2"),
  "key2" = c("K0","K1","K0","K1"),
  "A" = paste0('A',0:3),
  "B" = paste0('B',0:3)
)

df2 <- data.frame(
  "key1" = c("K0","K1","K1","K2"),
  "key2" = paste0('K',rep(0,4)),
  "C" = paste0('C',0:3),
```

```
  "D" = paste0('D',0:3)
)

full_join(df1,df2,by = c('key1','key2')) %>%
  na.omit(.)
```

109 数据重塑

题目：按照多列对数据进行合并

难度：☆☆

备注

只保存df1的数据

Python解法

```
pd.merge(df1, df2, how='left', on=['key1', 'key2'])
```

R语言解法

```
left_join(df1,df2,by = c('key1','key2'))
```

110 数据处理

题目：再次读取数据1并显示所有的列

难度：☆☆

备注

数据中由于列数较多中间列不显示

Python解法

```
df = pd.read_csv(r'C:\Users\chenx\Documents\Data Analysis\数据1.csv',enc
pd.set_option("display.max.columns", None)
```

R语言解法

```
df <- read_csv('数据1.csv', locale = locale(encoding = "GBK")) %>%
  print(width = Inf)
```

111 数据查找

题目：查找secondType与thirdType值相等的行号

难度：☆☆

Python解法

```
np.where(df.secondType == df.thirdType)
```

R语言解法

```
df %>%
  mutate(nrow = rownames(.)) %>%
  filter(secondType == thirdType) %>%
  select(nrow) %>%
  unlist()
```

112 数据查找

题目：查找薪资大于平均薪资的第三个数据

难度：☆☆☆

Python解法

```
np.argwhere(df['salary'] > df['salary'].mean())[2]
# array([5], dtype=int64)
```

R语言解法

```
df %>%
  mutate(nrow = rownames(.)) %>%
  filter(salary > mean(salary)) %>%
  select(nrow) %>%
  filter(row_number() == 3)
# # A tibble: 1 x 1
# nrow
# <chr>
# 1 6
```

113 数据计算

题目：将上一题数据的salary列开根号

难度：☆☆

Python解法

```
df[['salary']].apply(np.sqrt)
```

R语言解法

```
df %>%
  summarise(salary_sqrt = sqrt(salary))
```

114 数据处理

题目：将上一题数据的linestaion列按_拆分

难度：☆☆

Python解法

```
df['split'] = df['linestaion'].str.split('_')
```

R语言解法

```
df <- df %>%
  mutate(split = str_split(linestaion, '_'))
```


115 数据查看

题目：查看上一题数据中一共有多少列

难度：☆

Python解法

```
df.shape[1]
# 54
```

R语言解法

```
length(df)
# [1] 54
```

116 数据提取

题目：提取industryField列以'数据'开头的行

难度： ☆☆

Python解法

```
df[df['industryField'].str.startswith('数据')]
```

R语言解法

```
df[grep("^数据", df$industryField),]
```

117 数据计算

题目： 以salary score 和 positionID制作数据透视

难度： ☆☆☆

Python解法

```
pd.pivot_table(df,values=["salary","score"],index="positionId")
```

R语言解法

```
df <- df %>%
  group_by(positionId) %>%
  dplyr::summarise(salary = mean(salary),
            score = mean(score)) %>%
  as.data.frame(.)
rownames(df) <- NULL
tibble::column_to_rownames(df,var='positionId')
```


118 数据计算

题目： 同时对salary、score两列进行计算

难度： ☆☆☆

Python解法

```
df[["salary","score"]].agg([np.sum,np.mean,np.min])
```

R语言解法

```
res <- df %>%
  select(salary,score) %>%
  pivot_longer(c(salary,score),names_to = 'type',values_to = 'value') %>
  group_by(type) %>%
  summarise(sum = sum(value),mean = mean(value),min = min(value))
```

```
rownames(res) <- NULL

res %>%
  column_to_rownames('type') %>%
  t(.)
```

119 数据计算

题目：对不同列执行不同的计算

难度：☆☆☆

备注

对salary求平均，对score列求和

Python解法

```
df.agg({"salary":np.sum,"score":np.mean})
```

R语言解法

```
df %>%
  summarise(salary_sum = sum(salary),
            score_mean = mean(score))
```


120 数据计算

题目：计算并提取平均薪资最高的区

难度：☆☆☆☆

Python解法

```
df[['district','salary']].groupby(by='district').mean().sort_values(
    'salary',ascending=False).head(1)
```

R语言解法

```
df %>%
  group_by(district) %>%
  summarise(avg = mean(salary)) %>%
  arrange(desc(avg)) %>%
  filter(row_number() == 1)
```

以上就是玩转数据处理120题全部内容，如果能坚持走到这里的读者，我想你已经掌握了处理数据的常用操作，并且在之后的数据分析中碰到相关问题，希望武装了Pandas的你能够从容的解决！

另外我已将习题与源码整理成电子版，后台回复pandas即可下载，我们下个专题见，拜拜～

— The End —