import matplotlib.pyplot as plt

import pandas as pd

df=pd.read\_csv("/app/HDSBAT/eloisewu/jupyter/abc.csv",sep=',',header=0)

print(format(df))

pd.set\_option('display.width',200)

import matplotlib.pyplot as plt

import pandas as pd

import numpy as np

d = {'Name':pd.Series(['Tom','James','Ricky','Vin','Steve','Minsu','Jack']),

'Age':pd.Series([25,26,25,23,30,29,23]),

'Rating':pd.Series([4.23,3.24,3.98,2.56,3.20,4.6,3.8])}

df = pd.DataFrame(d)

print (df)

print(df.T)

print(df.axes)

print(df.dtypes)

print(df.empty)

print(df.ndim)

print(df.shape)

print(df.size)

print(df)

print(df.values)

print(df.head(2))

print(df.tail(2))

#arrays:([])

#list:[] [[1, 2], [3, 4]] list('abcde')

#dict:{'':1,'':2} /data={'name':(['a','b','c']),'age':arrays} arrays=([1,2,3])

#[{'a':1,'b':2},{'a':5,'b':6,'c':7}]

#data = {'one':pd.Series([1,2,3],index=['a','b','c']),'two':pd.Series([4,5,6,7],index=['a','b','c','d'])}

#series:

#dataframe

# pd.DataFrame([[1, 2], [3, 4]], columns = ['a','b'])

#+loc iloc ix

#append:

#df1=pd.DataFrame([[1,2],[3,4]],columns=['a','b'],index=[11,22])

#df2=pd.DataFrame([[5,6],[7,8]],columns=['c','d'],index=[33,'a'])

#print(df1.append(df2))

#print(df.drop('a'))

#---

#print(df.T)

#print(df.axes)

#print(df.dtypes)

#print(df.empty)

#print(df.ndim)

#print(df.shape)

#print(df.size)

#print(df)

#print(df.values)

#print(df.head(2))

#print(df.tail(2))

import pandas as pd

import numpy as np

d = {'Name':pd.Series(['Tom','James','Ricky','Vin','Steve','Minsu','Jack',

'Lee','David','Gasper','Betina','Andres']),

'Age':pd.Series([25,26,25,23,30,29,23,34,40,30,51,46]),

'Rating':pd.Series([4.23,3.24,3.98,2.56,3.20,4.6,3.8,3.78,2.98,4.80,4.10,3.65])}

df = pd.DataFrame(d)

print (df)

print (df.sum())

print(df.sum(1))

print(df.mean())

print(df.describe(include=['all']))

#DataFrame – “index”(axis=0，default)，”columns”(axis=1)

#s = pd.Series(np.random.randn(4))

#count()

#sum() sum(1)

#mean()

#median()

#mode()

#std()

#min()

#max()

#abs()

#prod()

#cumsum()

#cumprod()

#describe(include='all')

#describe(include=['object'])

#describe(include=['number'])

#表式函数应用

def adder(ele1,ele2):

return ele1+ele2

df.pipe(adder,2)

#行或列函数应用

df.apply(np.mean)

df.apply(np.mean,axis=1)

df.apply(lambda x: x.max() - x.min())

#元素函数应用

df.applymap(lambda x:x\*100)

import pandas as pd

import numpy as np

N=20

df = pd.DataFrame({

'A': pd.date\_range(start='2016-01-01',periods=N,freq='D'),

'x': np.linspace(0,stop=N-1,num=N),

'y': np.random.rand(N),

'C': np.random.choice(['Low','Medium','High'],N).tolist(),

'D': np.random.normal(100, 10, size=(N)).tolist()

})

print(df)

df\_reindexed = df.reindex(index=[0,2,5], columns=['A', 'C', 'B'])

print (df\_reindexed)

df1 = pd.DataFrame(np.random.randn(10,3),columns=['col1','col2','col3'])

df2 = pd.DataFrame(np.random.randn(7,3),columns=['col1','col2','col3'])

df1 = df1.reindex\_like(df2)

#Pandas 重新索引(Reindexing)

##重新排序现有数据以匹配一组新的标签。

#在没有标签数据的标签位置插入缺失值(NA)标记。

#df1 = df1.reindex\_like(df2)

#df\_reindexed = df.reindex(index=[0,2,5], columns=['A', 'C', 'B'])

print (df2.reindex\_like(df1)) = print (df2.reindex\_like(df1,method='bfill'))

print (df2.reindex\_like(df1,method='ffill')) = print (df2.reindex\_like(df1,method='nearest'))

print (df2.reindex\_like(df1,method='bfill'))

print (df2.reindex\_like(df1,method='nearest'))

print (df2.reindex\_like(df1,method='ffill',limit=2))

print (df1.rename(columns={'col1' : 'c1', 'col2' : 'c2'},

index = {0 : 'apple', 1 : 'banana', 2 : 'durian'}))

import pandas as pd

import numpy as np

df = pd.DataFrame(np.random.randn(4,3),columns=['col1','col2','col3'])

print(df)

#迭代

#Series – 值

#DataFrame – 列标签 for col in df:

# print (col)

#Pannel – 项目标签

#迭代行

#iteritems() – 迭代(key，value)对

#iterrows() – 将行迭代为(索引，系列)对

#itertuples() – 以namedtuples的形式迭代行

#for key,value in df.iteritems():

# print(key,value)

#for row\_index,row in df.iterrows():

# print(row\_index,row)

#for row in df.itertuples():

# print(row)

#permutation()

import pandas as pd

import numpy as np

unsorted\_df = pd.DataFrame({'col1':[2,1,1,1],'col2':[1,3,2,4]})

sorted\_df = unsorted\_df.sort\_values(by='col1' ,kind='quicksort')

print(unsorted\_df)

print(sorted\_df)

#sorted\_df=unsorted\_df.sort\_index()

#sorted\_df = unsorted\_df.sort\_index(ascending=False)

#sorted\_df=unsorted\_df.sort\_index(axis=1)

#sorted\_df = unsorted\_df.sort\_values(by=['col1','col2'])

#sorted\_df = unsorted\_df.sort\_values(by='col1')

#=sorted\_df = unsorted\_df.sort\_values(by='col1' ,kind='mergesort')

#=sorted\_df = unsorted\_df.sort\_values(by='col1' ,kind='heapsort')

#=sorted\_df = unsorted\_df.sort\_values(by='col1' ,kind='quicksort')

#nframe = pd.DataFrame(np.arange(25).reshape(5,5))

#new\_order = np.random.permutation(5)

#new\_order = [3,4,2]

#print(nframe.take(new\_order))

import pandas as pd

import numpy as np

s = {pd.Series(['Tom ', ' William Rick', 'John', 'Alber@t']),pd.Series(['Tom ', ' William Rick', 'John', 'Alber@t'])}

df = pd.DataFrame(s)

print (s)

print ("================= Split Pattern: ==================")

print (df.str.split(' '))

#忽略了NaN值

#将Series对象转换为String对象，然后执行该操作。

#索引？dataframe？

#s.str.lower()

#s.str.upper()

#s.str.len()

#s.str.strip() 帮助从两侧的系列/索引中的每个字符串中删除空格(包括换行符)。

#s.str.split(' ')

#s.str.cat(sep=' <=> ')

#s.str.get\_dummies() 返回具有单热编码值的数据帧(DataFrame)？

#s.str.contains(' ')

#s.str.replace('@','$')

#s.str.repeat(2)

#s.str.count('m')

#s.str. startswith ('T')

#s.str.endswith('t')

#s.str.find('e') -1表示元素中没有这样的模式可用 返回位置

#s.str.findall('e') 返回[] [e]

#s.str.swapcase()

#s.str.islower()

#s.str.isupper()

#s.str.isnumeric()

pd.get\_option("display.max\_rows")

pd.set\_option("display.max\_rows",80)

pd.reset\_option("display.max\_rows")

pd.describe\_option("display.max\_rows")

with pd.option\_context("display.max\_rows",10):

print(pd.get\_option("display.max\_rows")) #10

print(pd.get\_option("display.max\_rows")) #60

display.max\_rows 要显示的最大行数

display.max\_columns 要显示的最大列数

display.expand\_frame\_repr 显示数据帧以拉伸页面

display.max\_colwidth 显示最大列宽

display.precision 显示十进制数的精度

display.width 每一行的宽度（避免换行）

warnings.filterwarnings('ignore') # 关闭运行时的警告

np.set\_printoptions(linewidth=100, suppress=True) # 打印numpy时设置显示宽度，并且不用科学计数法显示

补充前面的

.loc() 基于标签

.iloc() 基于整数

.ix() 基于标签和整数，在0.20.0中已经不建议使用了

def.loc[index\_name, col\_name]

df.loc['a']

df.loc[['a', 'b', 'c']]

df.loc['c' : 'h']

df.loc[df.A>0.5]

df.loc[:,'A']

df.loc[:,['A','C']]

df.loc[['a','b','f','h'],['A','C']]

df.loc['a']>0

df.iloc[5]

df.iloc[[5, 1, 7]]

df.iloc[0:3]

df.iloc[np.array(df.A>0.5)]

df.iloc[list(df.A>0.5)]

df.iloc[:4]

df.iloc[1:5, 2:4]

df.iloc[[1, 3, 5], [1, 3]]

df.iloc[1:3, :]

df.iloc[:,1:3]

df.ix[:4]

df.ix[:,'A']

df['a', 'A'] X []操作只能输入一个维度，不能用逗号隔开输入两个维度。

.loc 和 .iloc只输入一维时选择的行，而[]选择的是列，并且必须使用列名。

df['A']

df[['A','B']]

df = pd.DataFrame(np.random.randn(8, 4), columns = ['A', 'B', 'C', 'D'])

print (df[2:2]) X

Empty DataFrame

Columns: [A, B, C, D]

Index: []

df.A

df.loc[lambda df:[0,1]]

df.iloc[lambda df:[0,1]]

import pandas as pd

import numpy as np

df = pd.DataFrame(np.random.randn(8, 4), columns = ['A', 'B', 'C', 'D'])

print (df[2:2])

print(df)

变化百分数(pct\_change())，协方差(cov())，相关系数(corr())，数据排名(rank())。

s.pct\_change()

df.pct\_change()

df.pct\_change(axis = 1)

s.rank()

默认为true的升序参数(ascending=True), 当设置为false(ascending=True)时，数据按照降序排序。

rank支持不同的tie-breaking方法：

average – 默认值，相同数据分配平均数

min – 相同数据分配最小等级

max – 相同数据分配最大等级

first – 相同数据根据出现在数组的顺序分配等级

count 非空数据的个数

sum 数据之和

mean 算术平均值

mad 平均绝对方差

median 中位数

min 最小值

max 最大值

mode 众数

abs 绝对值

prod 数组元素的乘积

std 标准差

var 方差

sem 标准误差

skew 偏差

kurt 样本值峰度

quantile 分位数

cumsum 累加

cumprod 累乘

cummax 累计最大值

cummin 累计最小值

cov() 协方差

corr() 相关系数

rank() 数据排名

pct\_change() 计算百分数变化

窗口函数

np.NaN = None

s.dropna()

print(ser[ser.notnull()])

df.dropna()#只要行或列有一个NaN元素，该行或列的全部元素都会被删除

df.dropna(how='all')#只删除所有元素均为NaN的行或列

df.fillna(0)

df.fillna({'ball':1,'mug':0,'pen':99})#依次指定列名称及要替换成的元素

dates = pd.date\_range('20130101', periods=6)

df = pd.DataFrame(np.random.randn(6, 4), index=dates, columns=list('ABCD'))

df2 = pd.DataFrame({'A': 1.,

'B': pd.Timestamp('20130102'),

'C': pd.Series(1, index=list(range(4)), dtype='float32'),

'D': np.array([3] \* 4, dtype='int32'),

'E': pd.Categorical(["test", "train", "test", "train"]),

'F': 'foo'})

A float64

B datetime64[ns]

C float32

D int32

E category

F object

dtype: object

In [12]: df2.<TAB> #按下tab键会自动补全所有的列名以及公共属性

df2.A df2.bool

df2.abs df2.boxplot

df2.add df2.C

df2.add\_prefix df2.clip

df2.add\_suffix df2.clip\_lower

df2.align df2.clip\_upper

df2.all df2.columns

df2.any df2.combine

df2.append df2.combine\_first

df2.apply df2.compound

df2.applymap df2.consolidate

df2.D........

dates = pd.date\_range('20130101', periods=6)

df = pd.DataFrame(np.random.randn(6, 4), index=dates, columns=list('ABCD'))

print(df.index)

print(df.columns)

freq='D'?

print(df)

print(df.values)

print(np.array(df))

print(df.describe())

print(df['A'])#选择一个列，产生一个“Series”

print(df[0:3])

print(df['20130102':'20130104'])

print(df.loc[dates[0]])

print(df.loc[:, ['A', 'B']])

print(df.loc['20130102':'20130104', ['A', 'B']])

print(df.loc['20130102', ['A', 'B']])

print(df.loc[dates[0], 'A'])#获取标量值

print(df.at[dates[0], 'A'])

print(df.iloc[3])

print(df.iloc[3:5, 0:2])

print(df.iloc[[1, 2, 4], [0, 2]])

print(df.iloc[:, 1:3])

print(df.iloc[1, 1])

print(df.iat[1, 1])

print(df[df.A > 0])

print(df[df > 0])

df2 = df.copy()

df2['E'] = ['one', 'one', 'two', 'three', 'four', 'three']

print(df2[df2['E'].isin(['two', 'four'])])

s1 = pd.Series([1, 2, 3, 4, 5, 6], index=pd.date\_range('20130102', periods=6))

print(s1)

df['F'] = s1

dates = pd.date\_range('20130101', periods=6)

df = pd.DataFrame(np.random.randn(6, 4), index=dates, columns=list('ABCD'))

s1 = pd.Series([1, 2, 3, 4, 5, 6], index=pd.date\_range('20130101', periods=6))

print(s1)

df['F'] = s1

df.at[dates[0], 'A'] = 0

df.iat[0, 1] = 0

df.loc[:, 'D'] = np.array([5] \* len(df))

df2 = df.copy()

df2[df2 > 0] = -df2

print(df.mean())

print(df.mean(1))

s = pd.Series([1, 3, 5, np.nan, 6, 8], index=dates).shift(2)

print(df.sub(s, axis='index')) #axis='columns'?

print(df.apply(np.cumsum)) #应用函数?

print(df.apply(lambda x: x.max() - x.min()))

s = pd.Series(np.random.randint(0, 7, size=10))#直方图化?

print(s.value\_counts())

print(s.str.lower())