**Pandas 数据分析**

日期：2017.6.5

|  |  |  |  |
| --- | --- | --- | --- |
| pd.DataFrame接收dict,list,可以传index,columns  ix[…]  mask  loc  argsort()  criteria：   |  |  | | --- | --- | | n. | （批评、判断等的） 标准，准则 |   closest to:  离…最近 | df = pd.DataFrame({'AAA' : [4,5,6,7], 'BBB' : [10,20,30,40],'CCC' : [100,50,-30,-50]}); df  df.ix[df.AAA>=5,'BBB']=-1;df  将AAA列大于5的行的BBB列设为-1  df.ix[df.AAA<5,['BBB','CCC']]=2000  将AAA列小于5的行的BBB和CCC列设为2000  df.ix[df.AAA < 5,['BBB','CCC']] = 2000; df  df\_mask = pd.DataFrame({'AAA' : [True] \* 4, 'BBB' : [False] \* 4,'CCC' : [True,False] \* 2})  df.where(df\_mask,-1000)  将df\_mask为真的位置的值保留，其他的设为-1000  df['logic']=np.where(df['AAA']>5,'high',low');df  新建logic列，将AAA大于5的行的此列的值设为high,否则设为low  d1=df[df.AAA<=5]  d2=df[df.AAA>5]  将df按AAA列值分割为两个  newseries = df.loc[(df['BBB'] < 25) & (df['CCC'] >= -40), 'AAA']; newseries  df.loc[(df['BBB'] > 25) | (df['CCC'] >= 75), 'AAA'] = 0.1; df  df.ix[(df.CCC-aValue).abs().argsort()] #求公式计算后相近的数据。 |
| DataFrame:以列名，此列的值方式进行赋值，并对值进行操作。  ix:按某列的值对行进行过滤，可以对过滤出来的行的其他列值进行修改或添加新列  mask:对应mask中的值进行修改  argsort求相近的值。 |

日期：2017.6.6

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| df.index.isin  binary  adj.二进制的; 二元的; 双重的，由两个东西组成的; 二态的;  ambiguity：   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | n. | 含糊; 意义不明确; 含糊的话，模棱两可的话; 可作两种或多种解释;  complement   |  |  | | --- | --- | | n. | 补充; 补足语; 补充物; 补集（数）; |  |  |  | | --- | --- | | vt. | 补足，补充; 补助; | | | df = pd.DataFrame({'AAA' : [4,5,6,7], 'BBB' : [10,20,30,40],'CCC' : [100,50,-30,-50]}); df  Crit1 = df.AAA <= 5.5  Crit2 = df.BBB == 10.0  Crit3 = df.CCC > -40.0  AllCrit = Crit1 & Crit2 & Crit3  CritList = [Crit1,Crit2,Crit3]  AllCrit = functools.reduce(**lambda** x,y: x & y, CritList)  df[AllCrit]  选择（Selection）  df[(df.AAA <= 6) & (df.index.isin([0,2,4]))]  data = {'AAA' : [4,5,6,7], 'BBB' : [10,20,30,40],'CCC' : [100,50,-30,-50]}  df = pd.DataFrame(data=data,index=['foo','bar','boo','kar']); df  df.loc['bar':'kar']  df.ix[0:3]  df.ix['bar':'kar'] #Same as .loc['bar':'kar']  df2 = pd.DataFrame(data=data,index=[1,2,3,4]); #Note index starts at 1.不是从0开始计数  df2.iloc[1:3] #Position-oriented，将按位置获取，iloc=indexloc(不包括3)index为位置  df2[1:3]与上一个意思一样  df2.loc[1:3]按index值进行选择= df2.ix[1:3]  df[~((df.AAA <= 6) & (df.index.isin([0,2,4])))]反向选择 |
| df.index.isin([0,2,4]):索引号为…的数据  df = pd.DataFrame(data=data,index=['foo','bar','boo','kar']);数据和索引分别定义  ~：反向选择。 |

日期：2017.6.6

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| date\_range  random.randn  transpose   |  |  | | --- | --- | | vt. | <数>移项; 使变位; <音>使变调; 变换顺序; |  |  |  | | --- | --- | | vi. | 进行变换; |  |  |  | | --- | --- | | n. | 转置阵; |   dimension   |  |  | | --- | --- | | n. | 尺寸; [复] 面积，范围; [物] 量纲; [数] 次元，度，维; | | Panels:3维数据容器，  rng = pd.date\_range('1/1/2013',periods=100,freq='D')  data = np.random.randn(100, 4) 产生4列100行随机数据  cols = ['A','B','C','D']  df1, df2, df3 = pd.DataFrame(data, rng, cols), pd.DataFrame(data, rng, cols), pd.DataFrame(data, rng, cols)  pf = pd.Panel({'df1':df1,'df2':df2,'df3':df3});pf  np.random.randn(2,5,4)生成2个5行4列的随机数  np.random.randn(2,5)生成2行5列的随机数  Pandas.Panel(data,items,major\_axis,minor\_axis)  items: 0轴， 每个项目对应其中的一个DataFrame  major\_axis: 1轴，wp.major\_xs(val)它是每个DataFrame的index（取出第几行）  minor\_axis: 2轴，wp.minor\_xs(val)它是每个DataFrame的column（取出第几列）  wp['Item1']  wp.major\_xs(wp.major\_axis[2])  wp.minor\_xs('C')  使用Panel.transpose(integer,integer,integer)可以重新分配Panel，integer取0，1，2分别对应轴0，1，2。  wp.transpose(2,0,1) |
| pd.date\_range('1/1/2013',periods=100,freq='D'):periods产生多少个数据，freq为数据产生周期，D为天，S为秒，M每月最后一天，H为小时 |

**INTRO TO DATA STRUCTURES**

日期：2017.6.6

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| comprehensive:   |  |  | | --- | --- | | adj. | 综合的; 广泛的; 有理解力的，悟性好的; [保险业] 总体担保的; | | n. | [常用复数] 专业综合考试; 综合学校; 综合性中学; |   overview:   |  |  |  | | --- | --- | --- | |  |  |  | | n. | 概观; 总的看法; 回顾，复习; | |   fundamental   |  |  | | --- | --- | | adj. | 基础的，基本的，根本的，重要的，原始的，主要的 |  |  |  | | --- | --- | | n. 原理，原则，基本，根本，基础; [乐] 基音; [物] 基频，基谐波; | | | axis | 英[ˈæksɪs] | | n.轴，轴线; [政] 轴心; 轴心国; [植] 茎轴; | | | collectively | 英[kə'lektɪvlɪ] |   adv.全体地，共同地;   |  |  | | --- | --- | | similarly | 英[ˈsɪmələli] | | adv.类似地; 相似地; | | | valid | 英[ˈvælɪd] | | adj.有效的; 有法律效力的; 正当的; 健全的; | | | argument | 英[ˈɑ:gjumənt] | | n.论据; 争论，争吵; [数] 幅角; 主题，情节; | | | Series  s = pd.Series(np.random.randn(5), index=['a', 'b', 'c', 'd', 'e'])  d = {'a' : 0., 'b' : 1., 'c' : 2.}  pd.Series(d)  pd.Series(d, index=['b', 'd', 'a'])没有的列不包括进来，不存在的以NaN（not a number）进行初始化。  pd.Series(5., index=['a', 'b', 'c', 'd', 'e'])：所有值均为5.0  ndarray 是一个多维的数组对象，具有矢量算术运算能力和复杂的广播能力，并具有执行速度快和节省空间的特点。  ndarray 的一个特点是同构：即其中所有元素的类型必须相同。  np.zeros(5)==array([0,0,0,0,0])  np.ones(4)==array([1,1,1,1])  np.arange(5)==array([0,1,2,3,4])  s[s > s.median()]取出大于中间值的数据  s[[4,3,1]]取子4，3，1位置的数据  s[‘a’],返回a的值  s[‘e’]=5,如果没有e则新建一个e  ‘e’ in s ：e是否在s中（index）  s.get(‘f’,5)如果f不存在则返回5，有则返回值。  s+s  s\*2  当key在各自的Series中时，（eg. s[1:]+s[:-1]）不是共同的key将被赋值为NaN  s = pd.Series(np.random.randn(5), name='something')  s.name:返回name  s.rename(“abc”) 重新命名 |
| flexibility: 英[ˌfleksə'bɪlətɪ]   |  | | --- | | n.柔度; 柔韧性，机动性，灵活性; 伸缩性; 可塑度; | | immense：英[ɪˈmens] | | adj.极大的，巨大的; 浩瀚的，无边际的; | |

日期：2017.6.6 **DataFrame**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | potentially | 英[pə'tenʃəlɪ] | | adv.潜在地; 可能地;  spreadsheet  英[ˈspredʃi:t]美[ˈsprɛdˌʃit]  n. 电子表格程序;  alternate 英[ɔ:lˈtɜ:nət]  美[ˈɔ:ltərnət]  adj. 交替的; 轮流的; 间隔的; 代替的;  vi. 交替; 轮流;  vt. 使交替; 使轮流; | | | d = {'one' : [1., 2., 3., 4.], 'two' : [4., 3., 2., 1.]}  df=pd.DataFrame(d, index=['a', 'b', 'c', 'd'])  data = np.zeros((2,), dtype=[('A', 'i4'),('B', 'f4'),('C', 'a10')])  产生两个元素的0矩阵，A列类型为4个字节，B列为单精度浮点数，C为10位字符，使用astype(float)可以改变类型  pd.DataFrame(data, columns=['C', 'A', 'B'])改变列的顺序。  data2 = [{'a': 1, 'b': 2}, {'a': 5, 'b': 10, 'c': 20}]  pd.DataFrame(data2)  Out[48]:  a b c  0 1 2 NaN   1. 5 10 20.0   通过元组字典可以自动创建多索引组织：  pd.DataFrame({('a', 'b'): {('A', 'B'): 1, ('A', 'C'): 2}, ('a', 'a'): {('A', 'C'): 3, ('A', 'B'):4}, ('a', 'c'): {('A', 'B'): 5, ('A', 'C'): 6}, ('b', 'a'): {('A', 'C'): 7, ('A', 'B'): 8}, ('b', 'b'): {('A', 'D'): 9, ('A', 'B'): 10}})  l=pd.DataFrame.from\_items([('A', [1, 2, 3]), ('B', [4, 5, 6])])  A B  0 1 4  1 2 5  2 3 6  l.assign(c=l['A']+l['B']),返回增加列c使用用A+B（原始数据不改变）  l[l.A>2]=l.query(‘A>2’)两个是一样的效果  l.assign(c=l['A']+l['B'],d=lambd x:x[‘A’]+x[‘B’]),可以同时设置两列。loc,iloc选择第几行数据。  df[:5].T 行列互换 |
| DataFrame用列标记不同类型的数据（每一列的数据可能不同）像电  子表格，数据库表，或序列的字典对象。 |

日期：2017.6.7

|  |  |
| --- | --- |
| interoperability  英['ɪntərɒpərə'bɪlətɪ]  美['ɪntərɒpərə'bɪlətɪ]  n. 互用性，协同工作的能力;  derived 英[dɪ'raɪvd]  美[dɪ'raɪvd]  vi. 衍生; 导出; 起源; 由来;  v. 得到( derive的过去式和过去分词 ); （从…中） 得到获得; 源于; （从…中） 提取;  partial 英[ˈpɑ:ʃl]  美[ˈpɑ:rʃl]  adj. 部分的; 偏爱的; 偏袒的; 钟爱的;  n. [数学] 偏微商;  particular  英[pəˈtɪkjələ(r)]  美[pərˈtɪkjələ(r)]  adj. 特别的; 详细的; 独有的; 挑剔的;  n. 特色，特点; （可分类，列举的） 项目; 详细情节，细情，细目; 某一事项;  slightly 英[ˈslaɪtli]  美[ˈslaɪtli]  adv. 轻微地，轻轻地; 细长地，苗条地; 〈罕〉轻蔑地; 粗;  arbitrary 英[ˈɑ:bɪtrəri]  美[ˈɑ:rbətreri]  adj. 乱; 随意的，任性的，随心所欲的; 主观的，武断的; 霸道的，专制的，专横的，独断独行的;  corresponds to：相当于  major 英[ˈmeɪdʒə(r)]  美[ˈmedʒɚ]  adj. 主要的; 重要的; 大调的; 主修的（课程）;  n. 主修科目; 大调; 陆军少校; 成年的;  vi. <美>主修，专攻; [美国英语][教育学]主修(in); 专攻; | DataFrame与Numpy互操作性  np.exp(df)  np.asarray(df)  df.T.dot(df)矩阵的倍增（自己乘自己再相加）  df.info()显示行列信息  df.to\_string()转换为string(列名为第一行，带\n)的  设置打印宽度  pd.set\_option('display.width', 40)在40列换行  pd.set\_option('display.max\_colwidth',100)最多打印100列，后面用…代替  df.列名 选择这一列打印  Panel:很少使用的3维数据结构  items: axis 0, each item corresponds to a DataFrame contained inside  major\_axis: axis 1, it is the index (rows) of each of the DataFrames  minor\_axis: axis 2, it is the columns of each of the DataFrames  wp = pd.Panel(np.random.randn(2, 5, 4), items=['Item1', 'Item2'],major\_axis=pd.date\_range('1/1/2000', periods=5), minor\_axis=['A', 'B', 'C', 'D'])  data = {'Item1' : pd.DataFrame(np.random.randn(4, 3)), 'Item2' : pd.DataFrame(np.random.randn(4, 2))}  p1=pd.Panel(data) |
| minor 英[ˈmaɪnə(r)]  美[ˈmaɪnɚ]  adj. 较小的，少数的，小…; 未成年的; [乐] 小调的，小音阶的; （两同姓男孩中） 年幼的;  n. 未成年人; 副修科目; 小公司; [逻辑学] 小前提;  vi. [主美国英语] 副修，选修，兼修; |

日期：

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| intersect 英[ˌɪntəˈsekt]  美[ˌɪntərˈsekt]  vt. 横断，横切，横穿;  vt. （指线条、道路等） 相交，交叉; | pd.Panel.from\_dict(data, orient='minor')  查看相应的items   |  |  |  |  |  | | --- | --- | --- | --- | --- | | Parameter | Default | Description | | | | intersect | False | drops elements whose indices do notalign | | | | orient | items | use minor to use DataFrames’ columns as panel items,用列名作为items。对混合型DataFrame有好处 | | | | Operation | | | Syntax | Result | | Select item | | | wp[item] | DataFrame | | Get slice at major\_axis label | | | wp.major\_xs(val) | DataFrame | | Get slice at minor\_axis label | | | wp.minor\_xs(val) | DataFrame |   wp.major\_xs(wp.major\_axis[2]),获取各Items第2行的值 |
|  |

日期：2017.6.7

|  |  |
| --- | --- |
| DataFrame：head(),tail()  DataFrame:sub()  essential 英[ɪˈsenʃl]  美[ɪˈsɛnʃəl]  adj. 必要的; 本质的; 基本的; 精华的;  n. 必需品; 基本要素; 必不可少的东西;  common to 英[ˈkɔmən tu:]  美[ˈkɑmən tu]  [词典] 共同的，共有的;  heterogeneous  英[ˌhetərəˈdʒi:niəs]  美[ˌhɛtərəˈdʒiniəs, -ˈdʒinjəs]  adj. 各种各样的; 成分混杂的;  accommodate 英[əˈkɒmədeɪt]  美[əˈkɑ:mədeɪt]  vt. 容纳; 使适应; 向…提供住处; 帮忙;  vi. [后面省去反身代词] 适应于; （眼）作调节，调节眼球的晶状体（使其变得适应不同距离的物体） ; 调解，调停;  flexible 英[ˈfleksəbl]  美[ˈflɛksəbəl]  adj. 灵活的; 柔韧的; 易弯曲的; 易被说服的;  demonstrate 英[ˈdemənstreɪt]  美[ˈdɛmənˌstret]  vt. 论证; 证明，证实; 显示，展示; 演示，说明;  simultaneously.  同时 | 生成测试数据：  index = pd.date\_range('1/1/2000', periods=8)  s = pd.Series(np.random.randn(5), index=['a', 'b', 'c', 'd', 'e'])  df = pd.DataFrame(np.random.randn(8, 3), index=index,columns=['A', 'B', 'C'])  wp = pd.Panel(np.random.randn(2, 5, 4), items=['Item1', 'Item2'], major\_axis=pd.date\_range('1/1/2000', periods=5), minor\_axis=['A', 'B', 'C', 'D'])  对确定的2进制数据和布尔操作使用numexpr和bottleneck库加快操作（推荐安装这两个库）  df.ix[1]选择第1行，df[‘列名’]  df.sub(row), sub(row,axis=’columns’),sub(row,axis=1) 每行都减去row的数据  axis代表列用什么  df.sub(column,axis=’index’)，每列都减去column的列值  df1.combine\_first(df2)，df1中为NaN时，用df2的值替换，都为NaN时设为NaN  combiner = lambda x, y: np.where(pd.isnull(x), y, x)  df1.combine(df2, combiner)  以上两句等于combine\_first  df.mean(0)对第0行计算平均数，df.mean()为对所有行分别计算平均数。  df.sum(0)对列进行求和，得到每列的和  df.sum(1)对行进行求和，得到每行的和 |
| carry out 英[ˈkæri aut]  美[ˈkæri aʊt]  [词典] 进行; 执行; 完成; 抬出去;  df.sub(column,axis=’index’)可以用于行驶时间的计算，出站时间减入站时间 |

日期：2017.6.8

|  |  |
| --- | --- |
|  | C:\Users\miao\AppData\Roaming\Tencent\Users\158132806\QQ\WinTemp\RichOle\GYDB7YP`(N4KV`1LT$$J7$W.png  series.describe()，返回count,mean,std数据  frame = pd.DataFrame(np.random.randn(1000, 5), columns=['a', 'b', 'c', 'd','e'])  frame[::2],frame[::5]，选择哪些行，行号能被2，5整除的行  series.describe(percentiles=[.05, .25, .75, .95]) 返回5%，25%。分位值 |
|  |

日期：2017.6.8

|  |  |
| --- | --- |
|  | df1.idxmin(axis=0)查找每列最小值的行号（0为行）  df1.idxmax(axis=1)查找每行最大值的列号（1为列）  s.value\_counts()s中各值计数统计，有多少相同的数据。(对速度进行计算，速度在时间上的分布；对入口出口进行统计，看车辆行驶方向)  s5.mode()众数，找出值最多的那个值。  df.appy(函数名)对df的值使用函数进行计算  l=pd.read\_csv('1.csv',header=0,encoding='GB2312')#读取文件注意编码  l['车辆入口站点'].mode() 取最大的数据计数  df1=l['车辆入口站点'].value\_counts() 取入口车辆数  df1[df1. 车辆入口站点>10000] 找出此列大于10000的行  l.head()  l.dropna()删除有NaN值的行  Aligning objects with each other with align  The align() method is the fastest way to simultaneously align two objects. It supports a join argument (related to  joining and merging):  • join='outer': take the union of the indexes (default)  • join='left': use the calling object’s index  • join='right': use the passed object’s index  • join='inner': intersect the indexes  df1.sort\_values(by=['two',’one’]),按by给的列名进行排序  s.nsmallest(3)取Series中最小的3个值  s.nlargest(3)取Series中最大的3个值  df.nlargest(3, 'a')取’a’列3个最大的行  df.nlargest(5, ['a', 'c']) |
|  |

日期：2017.6.12

|  |  |
| --- | --- |
| explicit 英[ɪkˈsplɪsɪt]  美[ɪkˈsplɪsɪt]  adj. 明确的，清楚的; 直言的; 详述的; 不隐瞒的;  seldom  英[ˈseldəm]美 [ˈsɛldəm]  adv.难得;很少，罕见;不大  To be clear：需要明确的是  the side effect of：副作用  potentially  英[pə'tenʃəlɪ]美 [pəˈtɛnʃəlɪ]  adv.  潜在地;可能地;<古>强有力地;权威地  gotchas陷阱，疑难杂症 | df1.columns=pd.MultiIndex.from\_tuples([('a','one'),('a','two'),('b','three')])  df1.sort\_values(by=('a','two'))  df3.astype('float32').dtypes  m = ['1.1', 2, 3]  pd.to\_numeric(m)  m = ['2016-07-09', datetime.datetime(2016, 3, 2)]  m = ['5us', pd.Timedelta('1day')]  pd.to\_timedelta(m)  df = pd.DataFrame([['2016-07-09', datetime.datetime(2016, 3, 2)]] \* 2,dtype='O')  df.apply(pd.to\_datetime)  df = pd.DataFrame([['1.1', 2, 3]] \* 2, dtype='O')//第一列串转为数值  df.apply(pd.to\_numeric)  dfi = df3.astype('int32')  dfi['E'] = 1  casted = dfi[dfi>0]  A B C E  0 NaN NaN NaN 1  1 NaN NaN 255.0 1  2 2.0 NaN 1.0 1  3 NaN NaN NaN 1  4 NaN NaN NaN 1  5 1.0 NaN NaN 1  6 NaN NaN NaN 1  7 NaN 1.0 NaN 1  df.select\_dtypes(include=[bool])  df.select\_dtypes(include=['bool'])  df.select\_dtypes(include=['number', 'bool'], exclude=['unsignedinteger']) |
| 多索引排序时需要列出所有索引号  if errors='coerce', these errors will be ignored  and pandas will convert problematic elements to pd.NaT (for datetime and timedelta) or np.nan (for numeric).  select 列出相关的数据,exclude为排除 |

日期：2017.6.12

|  |  |
| --- | --- |
|  | P499  s.str.lower()  s.str.upper()  s2.str.strip()去掉空格,rstrip,lstrip  s2.str.split('\_', expand=True) #expand=True返回一个DataFrame  s2.str.rsplit('\_', expand=True, n=1) #从右边开始分割，只分割第一个(n=1)  pd.Series(['a1', 'b2', 'c3']).str.extract('([ab])(\d)', expand=False)  #返回a,b开头，后面跟个数字的，转为a 数字 b数字 其它的为NaN NaN（正则表达式，匹配的转，只有a或b或数字就转）  s = pd.Series(['a', 'a|b', np.nan, 'a|c'])  s.str.get\_dummies(sep='|')  #对Series中的内容出现次数进行统计    p523 |
|  |

**索引和选择数据**

日期：2017.6.12

|  |  |
| --- | --- |
| invested  v.  投资，花费( invest的过去式和过去分词 );授予;（把资金）投入;投入（时间、精力等） | 数据选择的3种形式  loc(主要基于标签，但是也能使用boolean array)  iloc(基于整数位置的选择，从0到length-1,但是可以使用boolean array)  ix(支持混合的整数和标签的访问，以上两种)  Series s.loc[indexer]  DataFrame df.loc[row\_indexer,column\_indexer]  Panel p.loc[item\_indexer,major\_indexer,minor\_indexer]  x = pd.DataFrame({'x': [1, 2, 3], 'y': [3, 4, 5]})  x.iloc[1] = dict(x=9, y=99)  s=df[‘A’]  s[::2]#从第0个开始取，取到最后，每隔2个取一次数  s[::-1]#反序进行取数  s[::-2]#反序隔一取一  df[:5]取行（取几行）  df1.loc['2013-01-01':'2013-01-04']  df1.loc['2013-01-01','2013-01-04']  Select By Position  df[1:5,2:4]  选择1到5行的2到4列  x=list(‘abcde’)返回[‘a’,’b’,’c’,’d’,’e’] |
|  |

日期：2017.6.12

|  |  |
| --- | --- |
|  | Selection By Callable  df1=pd.DataFrame(np.random.randn(6,4),index=list('abcdef'),columns=list('ABCD'))  df1.loc[lambda df:df.A>0]  bb = pd.read\_csv('data/baseball.csv', index\_col='id')  (bb.groupby(['year', 'team']).sum()  l[l.车辆入口站点>500].sort\_values(by='出口车道',ascending=False)  s=pd.Series(np.random.randn(4))  s.sample(2)取两个样本（随机）  s.sample(n=6, replace=True)可以返回相同的值  根据权重来出样本  s = pd.Series([0,1,2,3,4,5])  example\_weights = [0, 0, 0.2, 0.2, 0.2, 0.4]  s.sample(n=3, weights=example\_weights)  dfi = pd.DataFrame(np.arange(6).reshape(3,2),columns=['A','B'])  reshape将6个数据转换为3\*2的矩阵  criterion = df2['a'].map(lambda x: x.startswith('t'))  a列以t开头的数据列出来。  df2[[x.startswith('t') for x in df2['a']]]与上一条效果一致  s = pd.Series(np.arange(5), index=np.arange(5)[::-1], dtype='int64')  s.isin([2,4,6])  s\_mi = pd.Series(np.arange(6),index=pd.MultiIndex.from\_product([[0, 1], ['a', 'b', 'c']])) MultiIndex可以用来进行多级别分隔  s\_mi.iloc[s\_mi.index.isin([(1, 'a'), (2, 'b'), (0, 'c')])],返回为isin为真的行 |
| groupby可以对车辆行驶进行统计  df.groupby(['车辆入口站点','车辆出口站点']).count()，对两站之间车辆流量进行统计 |

日期：2017.6.12

|  |  |
| --- | --- |
|  | df.where(df < 0, -df)  小于0的不变，大于0的取负数  df3 = pd.DataFrame({'A': [1, 2, 3],'B': [4, 5, 6],'C': [7, 8, 9]})  df3.where(lambda x: x > 4, lambda x: x + 10)  对大于4的值不变，小于4的值+10  df3.where(df3>3)大于3的值列出，其它为NaN  df3，where(条件为真的不变，不符合条件的操作)  df3.mask(df3>3)大于3的值屏蔽，小于3的列出来  df[(df.a < df.b) & (df.b < df.c)]  df.query('(a < b) & (b < c)') #两句效果一样  找出a列小于b列，b列小于c列的值  df.query('index < b < c')  index为索引，如果索引名为a且有a列，则用a表示列，index表示索引。  n = 10  colors = np.random.choice(['red', 'green'], size=n)  foods = np.random.choice(['eggs', 'ham'], size=n)  index = pd.MultiIndex.from\_arrays([colors, foods], names=['color', 'food'])  df = pd.DataFrame(np.random.randn(n, 2), index=index)  df.query('color=="red" and food=="ham"')  df[(df.a < df.b) & (df.b < df.c)]#纯Python  df.query('(a < b) & (b < c)') #query形式  ilevel\_0就是“index level 0”,第一个索引 |
|  |

日期：2017.6.12

|  |  |
| --- | --- |
|  | df.query('a in b')  df.a.isin(df.b)]  df.query('a not in b')  df[~df.a.isin(df.b)]  df.query('b == ["a", "b", "c"]')  df[df.b.isin(["a", "b", "c"])]  df.query('c == [1, 2]')  df.query('c != [1, 2]')  df.query('[1, 2] in c')  df2.duplicated('a')  df2.duplicated('a', keep='last')  • keep='first' (default): mark / drop duplicates except for the first occurrence.  • keep='last': mark / drop duplicates except for the last occurrence.  • keep=False: mark / drop all duplicates.  dflookup = pd.DataFrame(np.random.rand(20,4), columns = ['A','B','C','D'])  dflookup.lookup(list(range(0,10,2)), ['B','C','A','B','D'])  索引为0,2,4,6,8列为BCABD的值  index = pd.Index(list(range(5)), name='rows')  columns = pd.Index(['A', 'B', 'C'], name='cols')  df = pd.DataFrame(np.random.randn(5, 3), index=index, columns=columns)  df |
|  |

日期：2017.6.12

|  |  |
| --- | --- |
|  | frame = data.set\_index('c', drop=False),#按c列索引，数据中还要保留c列  frame = frame.set\_index(['a', 'b'], append=True)，添加ab两列为索引，还要保留原来的索引  data.reset\_index()#恢复默认索引  frame.reset\_index(level=1)  删除哪个级别的索引，从0开始算  索引行注意，如果有错误数据将报错。  data.index = index  可以自己建一个索引，然后赋值给DataFrame  dfmi = pd.DataFrame([list('abcd'),list('efgh'),list('ijkl'),  list('mnop')],columns=pd.MultiIndex.from\_product  ([['one','two'],['first','second']]))  dfmi['one']['first']  d.rank(0)，按行对第列数据统计排在第几位  d.rand(1),数据在本行中排第几位  df.groupby('车辆入口站点').get\_group('1171707')  得到值为’1171707’这一组的值 |
|  |

日期：2017.6.12 P681

|  |  |
| --- | --- |
|  | result = pd.concat([df1, df4], axis=1)    C:\Users\miao\AppData\Roaming\Tencent\Users\158132806\QQ\WinTemp\RichOle\93M7P0{Z`ATK65@N[U7ICRB.png  result = pd.concat([df1, df4], axis=1, join='inner') |
|  |

日期：2017.6.12

|  |  |
| --- | --- |
|  | result = pd.concat([df1, df4], axis=1, join\_axes=[df1.index])      以df1.index为标准进行数据整合  result = df1.append([df2, df3]) |
| append:存在不相同的列时，将没有值的列设为NaN |

日期：2017.6.12

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | import pandas as pd  import numpy as np  s3 = pd.Series([0, 1, 2, 3], name='foo')  s4 = pd.Series([0, 1, 2, 3])  s5 = pd.Series([0, 1, 4, 5])  pd.concat([s3, s4, s5], axis=1, keys=['red','blue','yellow'])   |  | **red** | **blue** | **yellow** | | --- | --- | --- | --- | | **0** | 0 | 0 | 0 | | **1** | 1 | 1 | 1 | | **2** | 2 | 2 | 4 | | **3** | 3 | 3 | 5 |   s2 = pd.Series(['X0', 'X1', 'X2', 'X3'], index=['A', 'B', 'C', 'D'])  result = df1.append(s2, ignore\_index=True)    将一个Series添加为DataFrame的一行 |
|  |

日期：2017.6.12

|  |  |
| --- | --- |
|  | pieces = {'x': df1, 'y': df2, 'z': df3}  result = pd.concat(pieces) |
|  |

日期：2017.6.12

|  |  |
| --- | --- |
|  | pd.merge(left, right, how='inner', on=None, left\_on=None, right\_on=None,left\_index=False, right\_index=False, sort=True,  suffixes=('\_x', '\_y'), copy=True, indicator=False)  result = pd.merge(left, right, on='key')      result = pd.merge(left, right, on=['key1', 'key2']) |
|  |

日期：2017.6.12

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | df1 = pd.DataFrame({'col1': [0, 1], 'col\_left':['a', 'b']})  df2 = pd.DataFrame({'col1': [1, 2, 2],'col\_right':[2, 2, 2]})  pd.merge(df1, df2, on='col1', how='outer', indicator=True)   |  | **col1** | **col\_left** | **col\_right** | **\_merge** | | --- | --- | --- | --- | --- | | **0** | 0 | a | NaN | left\_only | | **1** | 1 | b | 2.0 | both | | **2** | 2 | NaN | 2.0 | right\_only | | **3** | 2 | NaN | 2.0 | right\_only | |  |  |  |  |  |   \_merge列显示数据在那边存在 |
|  |

日期：2017.6.12

|  |  |
| --- | --- |
|  | pd.Timestamp(datetime.datetime(2012, 5, 1))  pd.Timestamp('2012-05-01')  两个得到Timestamp('2012-05-01 00:00:00')  pd.to\_datetime(pd.Series(['Jul 31, 2009', '2010-01-10', None]))  0 2009-07-31  1 2010-01-10  2 NaT  dtype: datetime64[ns]  pd.to\_datetime(['04-01-2012 10:00'], dayfirst=True)  dayfirst日月年改为年月日（欧式风格）  df = pd.DataFrame({'year': [2015, 2016],  'month': [2, 3],  'day': [4, 5],  'hour': [2, 3]})  pd.to\_datetime(df)  0 2015-02-04 02:00:00  1 2016-03-05 03:00:00  dtype: datetime64[ns]  index = pd.date\_range('2000-1-1', periods=1000, freq='M')  产生由2001年1月开始每月最后一天的日期，共1000个  date\_range为日历日期，bdate\_range为工作日期  dft[datetime(2013, 1, 1):datetime(2013,2,28)]  得到2013-1-1到2013-2-28之间的数据  dft[datetime(2013, 1, 1, 10, 12, 0):datetime(2013, 2, 28, 10, 12, 0)]  得到两个时间之间的数据 |
|  |

日期：2017.7.5

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| leap year闰年 | |  |  | | --- | --- | | Property | Description | | year | The year of the datetime | | month | The month of the datetime | | day | The days of the datetime | | hour | The hour of the datetime | | minute | The minutes of the datetime | | second | The seconds of the datetime | | microsecond | The microseconds of the datetime | | nanosecond | The nanoseconds of the datetime | | date | Returns datetime.date (does not contain timezone information) | | time | Returns datetime.time (does not contain timezone information) | | dayofyear | The ordinal day of year | | weekofyear | The week ordinal of the year | | week | The week ordinal of the year | | dayofweek | The numer of the day of the week with Monday=0, Sunday=6 | | weekday | The number of the day of the week with Monday=0, Sunday=6 | | weekday\_name | The name of the day in a week (ex: Friday) | | quarter | Quarter of the date: Jan=Mar = 1, Apr-Jun = 2, etc. | | days\_in\_month | The number of days in the month of the datetime | | is\_month\_start | Logical indicating if first day of month (defined by frequency) | | is\_month\_end | Logical indicating if last day of month (defined by frequency) | | is\_quarter\_start | Logical indicating if first day of quarter (defined by frequency) | | is\_quarter\_end | Logical indicating if last day of quarter (defined by frequency) | | is\_year\_start | Logical indicating if first day of year (defined by frequency) | | is\_year\_end | Logical indicating if last day of year (defined by frequency) | | is\_leap\_year | Logical indicating if the date belongs to a leap year |   s.dt.year  s.dt.is\_leap\_year  进行数据的获得 |
|  |

日期：2017.6.12

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| experimental  英[ɪkˌsperɪˈmentl]  美[ɪkˌspɛrəˈmɛntl]  adj. 实验的; 根据实验的; 试验性的;  n. 试验性的东西;  semi month半月，第15天和月底 | |  |  | | --- | --- | | Alias | Description | | B | business day frequency | | C | custom business day frequency (experimental) | | D | calendar day frequency | | W | weekly frequency | | M | month end frequency | | SM | semi-month end frequency (15th and end of month) | | BM | business month end frequency | | CBM | custom business month end frequency | | MS | month start frequency | | SMS | semi-month start frequency (1st and 15th) | | BMS | business month start frequency | | CBMS | custom business month start frequency | | Q | quarter end frequency | | BQ | business quarter endfrequency | | QS | quarter start frequency | | BQS | business quarter start frequency | | A | year end frequency | | BA | business year end frequency | | AS | year start frequency | | BAS | business year start frequency | | BH | business hour frequency | | H | hourly frequency | | T, min | minutely frequency | | S | secondly frequency | | L, ms | milliseconds | | U, us | microseconds | | N | nanoseconds | |
|  |

日期：2017.7.5

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | d = datetime.datetime(2008, 8, 18, 9, 0)  d+pd.DateOffset(months=4,days=5)  Timestamp('2008-12-23 09:00:00')   |  |  | | --- | --- | | Class name | Description | | DateOffset | Generic offset class, defaults to 1 calendar day | | BDay | business day (weekday) | | CDay | custom business day (experimental) | | Week | one week, optionally anchored on a day of the week | | WeekOfMonth | the x-th day of the y-th week of each month | | LastWeekOfMonth | the x-th day of the last week of each month | | MonthEnd | calendar month end | | MonthBegin | calendar month begin | | BMonthEnd | business month end | | BMonthBegin | business month begin | | CBMonthEnd | custom business month end | | CBMonthBegin | custom business month begin | | Continued on next page | |   Table 20.1 – continued from previous page   |  |  | | --- | --- | | Class name | Description | | SemiMonthEnd | 15th (or other day\_of\_month) and calendar month end | | SemiMonthBegin | 15th (or other day\_of\_month) and calendar month begin | | QuarterEnd | calendar quarter end | | QuarterBegin | calendar quarter begin | | BQuarterEnd | business quarter end | | BQuarterBegin | business quarter begin | | FY5253Quarter | retail (aka 52-53 week) quarter | | YearEnd | calendar year end | | YearBegin | calendar year begin | | BYearEnd | business year end | | BYearBegin | business year begin | | FY5253 | retail (aka 52-53 week) year | | BusinessHour | business hour | | CustomBusinessHour | custom business hour | | Hour | one hour | | Minute | one minute | | Second | one second | | Milli | one millisecond | | Micro | one microsecond | | Nano | one nanosecond | |
| from pandas.tseries.offsets import \*  引用此句后可以直接调用DateOffset  d-5\*BDay()d的往前第5个工作日  d + BMonthEnd()d日期所在月的最后一个工作日  d=datetime.datetime(2008, 8, 18, 9, 0)  offset = BMonthEnd()  offset.rollforward(d) 求d日期后面的月底工作日  offset.rollback(d) 求d日期之前的月底工作日 |

日期：2017.7.5

|  |  |
| --- | --- |
|  | day = Day()  day.apply(pd.Timestamp('2014-01-01 09:00'))  apply默认增加一天（2014-01-02 09:00）  day = Day(normalize=True)时间变为2014-01-02 00:00:00  day.apply(pd.Timestamp('2014-01-01 09:00'))  hour = Hour()  hour.apply(pd.Timestamp('2014-01-01 22:00'))  hour = Hour(normalize=True)  hour.apply(pd.Timestamp('2014-01-01 22:00'))  hour.apply(pd.Timestamp('2014-01-01 23:00'))  **Custom Business Days (Experimental)**  **from pandas.tseries.offsets import** CustomBusinessDay  weekmask\_egypt = 'Sun Mon Tue Wed Thu'  holidays = ['2012-05-01', datetime(2013, 5, 1), np.datetime64('2014-05-01')]  bday\_egypt = CustomBusinessDay(holidays=holidays, weekmask=weekmask\_egypt)  dt = datetime(2013, 4, 30)  dt + 2 \* bday\_egypt  dts = pd.date\_range(dt, periods=5, freq=bday\_egypt)  pd.Series(dts.weekday, dts).map(pd.Series('Mon Tue Wed Thu Fri Sat Sun'.split())) |
|  |

日期：2017.7.5 P785

|  |  |
| --- | --- |
|  | s = pd.Series(pd.date\_range('2012-1-1', periods=3, freq='D'))  td = pd.Series([ pd.Timedelta(days=i) for i in range(3) ])  df = pd.DataFrame(dict(A = s, B = td))  df['C']=df['A']+df['B']  C:\Users\miao\AppData\Roaming\Tencent\Users\158132806\QQ\WinTemp\RichOle\IUOY3}B~OK(_]SJB5]]2M)U.png  df = pd.DataFrame(np.random.randn(1000, 4), index=ts.index, columns=list('ABCD'))  df = df.cumsum()  plt.figure()  %matplotlib inline#显示图形  df.plot() |
|  |

日期：2017.6.12

|  |  |
| --- | --- |
|  |  |
|  |

日期：2017.6.12

|  |  |
| --- | --- |
|  |  |
|  |

日期：2017.6.12

|  |  |
| --- | --- |
|  |  |
|  |