第九章
数据分析与可视化

第九章 数据分析与可视化

本章重点介绍三个程序包numpy、pandas和matpoltlib的用法。这三个程序包均为Python语言的外部扩展包，使用它们前需要安装这些包。在PyCharm中，单击菜单“文件 | 设置 | 项目: MyPythonPrj | Python解释器”，这里的“MyPythonPrj”为本书所使用的Python工程名。然后，单击显示的页面的左上方的“+”号安装这三个程序包的最新版本。当前，numpy的版本为1.23.1；pandas的版本为1.4.3；matplotlib的版本为3.5.2。

第九章 数据分析与可视化

程序包numpy实现了一种新的数据类型——数组，与C语言意义上的数组概念相同，即包含相同类型元素的一组数据（事实上，这种数据类型类似于Python语言内置的“列表”，可以存储不同类型的数据，例如，同时存储整数和字符串，但是numpy提供的函数主要是针对同类型数据），同时，numpy基于数组提供了大量的函数，包括数学函数、统计函数、伪随机数函数和字符串函数等，使得numpy应用广泛。此外，numpy的处理速度快，并且是pandas等外部程序包的基础。

程序包pandas实现了两种新的数据类型，即Series（序列）和DataFrame（数据框架），这两种数据类型的优势在于可以处理数组中的缺失值。pandas提供了基于Series和DataFrame数据对象的大量统计函数。pandas包主要针对数据统计和大数据处理等应用。

程序包matplotlib是一个二维绘图库，可以绘制各种数学函数图形和金融数据图形，是Python语言中数据可视化的重要工具，其特点在于绘图函数简单易用，借用了MATLAB软件中的绘图函数的名称，生成的图形可以用于科技论文中。

第九章 数据分析与可视化

本章将分为三节内容来介绍程序包numpy、pandas和matpoltlib的用法：

9.1 程序包numpy

程序包numpy的算法基于其自定义的数据类型——数组，一般地，认为数组具有相同类型的元素。元素类型可以为字符串和各种数据类型，例如，

```
import numpy as np   #装载包numpy，并赋以别名np
    arr1=np.array([1,2,3],dtype='int_')
    print(arr1)
    print(arr1.dtype)
```

将创建一个数组arr1，其元素为“1, 2, 3”，“dtype”用于指定数组元素的类型，这里的“int_”为默认的整型，因计算机而异，可能为32位整型int32，或64位整型int64。“print(arr1)”输出数组arr1，得到“[1, 2, 3]”；“print(arr1.dtype)”输出数组（元素）的类型。

9.1 程序包numpy

一般地，使用默认的元素数据类型创建数组时不需要指定“dtype”。默认的元素数据类型有“int_”、“float_”、“complex_”，分别表示32位或64位有符号整型、64位浮点型（IEEE754格式，其中，1个符号位、11个指数位、52个尾数位）和128位复数型（其中，实部和虚部各占64位）。其他的常用数值型数据类型包括8位有符号整型（int8）、8位无符号整型（uint8）、16位有符号整型（int16）、16位无符号整型（uint16）、32位有符号整型（int32）、32位无符号整型（uint32）、64位有符号整型（int64）、64位无符号整型（uint64）、单精度浮点数（float32）、双精度浮点数（float64）、复数complex128（实部和虚部分别为64位双精度浮点数）等。

9.1.1 数组创建

一般地，认为程序包numpy的所有方法均基于其自定义数据类型——数组，所以，使用包numpy的方法前，必须定义数组。

创建数组的方法有以下四种：

（1）将列表转化为数组

包numpy的array方法可以将列表转化为数组。对于一维列表，将被转化一维数组；对于二维的嵌套列表，要求列表中每个子列表的元素个数必须相同，这样的嵌套列表被转化为二维数组；对于高维的嵌套列表，同样要求同级别的子列表的元素个数相同，这样的高维嵌套列表被转化为高维数组。

（2）使用程序包numpy的内置方法生成特殊数组

程序包numpy具有生成特殊数组的内置方法，这些方法包括zeros、zeros_like、ones、ones_like、empty等。其中，zeros方法可以生成元素全为0的数组，zeros生成二维以上的数组时，使用元组作为参数，例如：zeros((3,4))生成3行4列的数组，元素均为0。zeros_like方法的参数为一个数组，将生成与该数组相同结构的数组，其元素均为0。ones和ones_like方法分别与zeros和zeros_like方法相似，只是生成的数组元素均为1。empty方法生成一个数组，主要是为新生成的数组开辟存储空间（其元素值不定），为后续使用该数组的方法服务。

9.1.1 数组创建

（3）使用程序包numpy的内置方法生成规则数组

生成规则数组的方法主要有arrange和linspace。其中，arrange方法的语法为：arrange(起始值, 终止值（不含）, 步长)。例如，arrange(1,3,0.5)将得到“[1.0 1.5 2.0 2.5]”。

linspace方法的语法为：linspace(起始值, 终止值（含）, 分隔点数)。例如，linspace(1,3,5)得到“[1.0 1.5 2.0 2.5 3.0]”。

可见，arrange和linspace均用于生成等差数列形式的数组。

（4）使用程序包numpy的内置类random生成伪随机形式的数组

程序包numpy的内置类具有rand、randn和uniform等方法，可以生成伪随机数组。其中，rand用于生成[0,1)上均匀分布的伪随机数组，例如，rand(3,2)生成一个3行2列的[0,1)上均匀分布的伪随机数组。randn生成均值为0、方差为1的伪随机数组，例如，randn(3,2)生成一个3行2列的正态分布的伪随机数组。uniform(左边界, 右边界, 数组大小)用于生成[左边界, 右边界)上的均匀分布的数组，数组大小由参数“数组大小”决定。例如，uniform(1,10,5)生成长度为5的[1,10)上均匀分布的伪随机数组；uniform(1,10,(3,2))生成3行2列的数组，每个数组元素服从[1,10)上的均匀分布。

9.1.1 数组创建

下面的实例介绍了上述创建数组的方法：（第一部分）

```
1    import numpy as np
2    if __name__=='__main__':
3        arr1=np.array([1,2,3,4,5])
4        arr2=np.array([[1,2,4],[4,5,6]])
5        print(f'arr1={arr1}')
6        print(f'arr2=\n{arr2}')
7        arr3=np.ones(5)
8        arr4=np.ones((3,4))
9        print(f'arr3={arr3}')
10       print(f'arr4=\n{arr4}')
11       arr5=np.zeros_like(arr2)
12       arr6=np.ones_like(arr2)
13       print(f'arr5=\n{arr5}')
14       print(f'arr6=\n{arr6}')
```

第1行装载numpy程序包，并赋以别名np。

第3行将列表“[1,2,3,4,5]”转化为一维数组，赋给arr1。第4行将嵌套列表“[[1,2,4],[4,5,6]]”转化为2行3列的二维数组，赋给arr2。

第5、6行依次输出数组arr1和arr2。

第7行调用ones方法生成长度为5的数组，其元素均为1。

第8行生成3行4列的数组，其元素均为1。

第9、10行依次输出数组arr3和arr4。

第11行生成与arr2同样大小的数组，其元素均为0。

第12行生成与arr2同样大小的数组，其元素均为1。

第13、14行依次输出数组arr5和arr6。

9.1.1 数组创建

（第二部分）

```
14          print(f'arr6
15          arr7=np.ar
16          arr8=np.li
17          print(f'arr7
18          print(f'arr8
19          np.random
20          arr9=np.ra
21          arr10=np.r
22          arr11=np.r
23          np.random
24          print(f'arr9
25          print(f'arr1
26          print(f'arr11={arr11}')
```

运行: zym0901

```
C:\ZYWork\ZYPython\MyPythonPrj\venv\Scripts\python.exe C:/ZYWork/ZYPy
arr1=[1 2 3 4 5]
arr2=
[[1 2 4]
 [4 5 6]]
arr3=[1. 1. 1. 1. 1.]
arr4=
[[1. 1. 1. 1.]
 [1. 1. 1. 1.]
 [1. 1. 1. 1.]]
arr5=
[[0 0 0]
 [0 0 0]]
arr6=
[[1 1 1]
 [1 1 1]]
arr7=[1.  1.5 2.  2.5]
arr8=[1.  1.5 2.  2.5 3. ]
arr9=
[[0.62247643 0.68865483]
 [0.47100365 0.93173406]
 [0.58617439 0.3081514 ]]
arr10=[ 0.8721242  -0.3913014   0.51987839  0.42458751]
arr11=[4.52950308 3.36351479 4.92264861 9.06172918 6.38704275]
```

成等差数列形式的数组arr
到3（不含）。
生成1至3间等差排列的长度
18行依次输出arr7和arr

器的“种子”为29979245
置一个种子值的原因在于，
生成的伪随机数序列相同。
“种子”，伪随机数发生器
”作为种子，从而第20~22
时都将得到不同的值。第23
为伪随机数发生器的“种子

布的3行2列的数组arr9。第
正态分布的伪随机数组arr1
0。第22行生成长度为3的服从1至10（不含）间均匀分布
的伪随机数组arr11。第24~26行依次输出数组arr9、
arr10和arr11。

9.1.2 数组元素访问

数组元素的访问与“列表”元素的访问方法相同。对于一个一维数组arr1，其元素索引号从左向右为0至“元素总个数减1”；从右向左为-1至“元素总个数的相反数”。访问arr1中元素的方法主要有两种：其一，arr1[n]访问数组arr1中索引号为n的元素；其二，arr1[m:n:k]访问数组arr1中索引号自m至n（不含）步长为k的元素，省略k时表示默认步长为1，省略m时表示起始索引号为0，省略n时表示n取为数组的最后一个元素的索引号加1的值。arr1.shape将返回一个元组，类似于“（7,）”，这里的“7”表示arr1的元素个数。arr1.size返回一个数，表示arr1的元素总个数。arr1.ndim返回数值1，表示arr1为一维数组。

9.1.2 数组元素访问

对于一个二维数组arr2，其行索引号从0至“总行数减1”（或从“总行数的相反数”至-1），其列索引号从0至“总列数减1”（或从“总列数的相反数”至-1）。访问arr2中元素的方法为：其一，arr2[m1,m2]访问数组arr2中行索引号为m1、列索引号为m2的元素；其二，arr2[m1:n1:k1,m2:n2:k2]访问数组arr2中行索引号为m1至n1（不含）步长为k1、列索引号为m2至n2（不含）步长为k2的全部元素，步长k1或k2省略时使用缺省值1；起始值m1或m2省略时使用索引号0；终止值n1或n2省略时使用行或列的最大索引号加上1的值。arr2.shape返回一个元组，包括arr2的行数和列数；arr2.ndim返回值2，表示arr2为一个二维数组；arr2.size返回一个数，表示arr2中总的元素个数。

9.1.2 数组元素访问

下面的实例展示了数组元数访问方法：

```
1   import numpy as np
2   if __name__=='__main__':
3       arr1=np.array([[1,2,3
4       arr2=np.array([[2,4]
5       arr3=np.array([[1,2,3
6       print(f'arr1=\n{arr1}
7       print(f'arr1 shape: {a
8       print(f'arr2=\n{arr2}
9       print(f'arr3=\n{arr3}
10      print(f'arr1[0,0]={ar
11      print(f'arr2[1,:]={arr
12      print(f'arr3[0,0:-1:2]
13      arr3[0,0]=21
14      print(f'arr3=\n{arr3}
15      arr3[0,:]=[11,12,13,1
16      print(f'arr3=\n{arr3}
17      arr4=np.array([1,2,3
18      print(f'arr4= {arr4}')
19      print(f'arr4 shape: {arr4.shape},arr4 size: {arr
```

```
运行:  zym0902
C:\ZYWork\ZYPython\MyPythonPrj\venv\Scripts\python.exe C:/ZYWork
arr1=
[[1 2 3]
 [4 5 6]]
arr1 shape: (2, 3),arr1 size: 6,arr1 ndim: 2
arr2=
[[2 4]
 [3 5]
 [4 6]]
arr3=
[[ 1  2  3  4  5]
 [ 6  7  8  9 10]]
arr1[0,0]=1,arr1[1,-1]=6
arr2[1,:]=[3 5],arr2[:,0]=[2 3 4]
arr3[0,0:-1:2]=[1 3],arr3[1,2:-1]=[8 9]
arr3=
[[21  2  3  4  5]
 [ 6  7  8  9 10]]
arr3=
[[11 12 13 14 15]
 [ 6  7  8  9 10]]
arr4= [1 2 3 4 5 6 7]
arr4 shape: (7,),arr4 size: 7,arr4 ndim: 1
```

赋以别外np。

第4行“创建3行2列的数组arr2。

第7行“输出arr1的形状、大小和
元素；第9行输出数组arr3的全部元

的元素以及第1行最后一列的元素。
部元素以及第0列的全部元素。
依步长2至最后一列（不含）的元
含）的元素。
元素为21。
此时，其第0行第0列的元素为21。
为列表“[11,12,13,14,15]”。
此时，其第0行的元素为“[11 12
以空格为分界符。

大小和维数，这里，arr4的形状为“
（7，）”，大小为7，维数为1。

9.1.3 矩阵运算

一般地，二维规则数组称为矩阵，即矩阵的每一行具有相同个数的元素。常用的矩阵运算有：

（1）单个数值与矩阵的和。此时，将单个数值加到矩阵的每个元素中。

（2）单个数值与矩阵的积，称为数乘。此时，将单个数值乘以矩阵的每个元素。

（3）两个矩阵的和。要求参与相加运算的两个矩阵具有相同的行、列数。此时，两个矩阵对应位置上的元素取和。

（4）两个矩阵的数量积。要求参与相乘运算的两个矩阵具有相同的行、列数。此时，两个矩阵对应位置上的元素相乘。

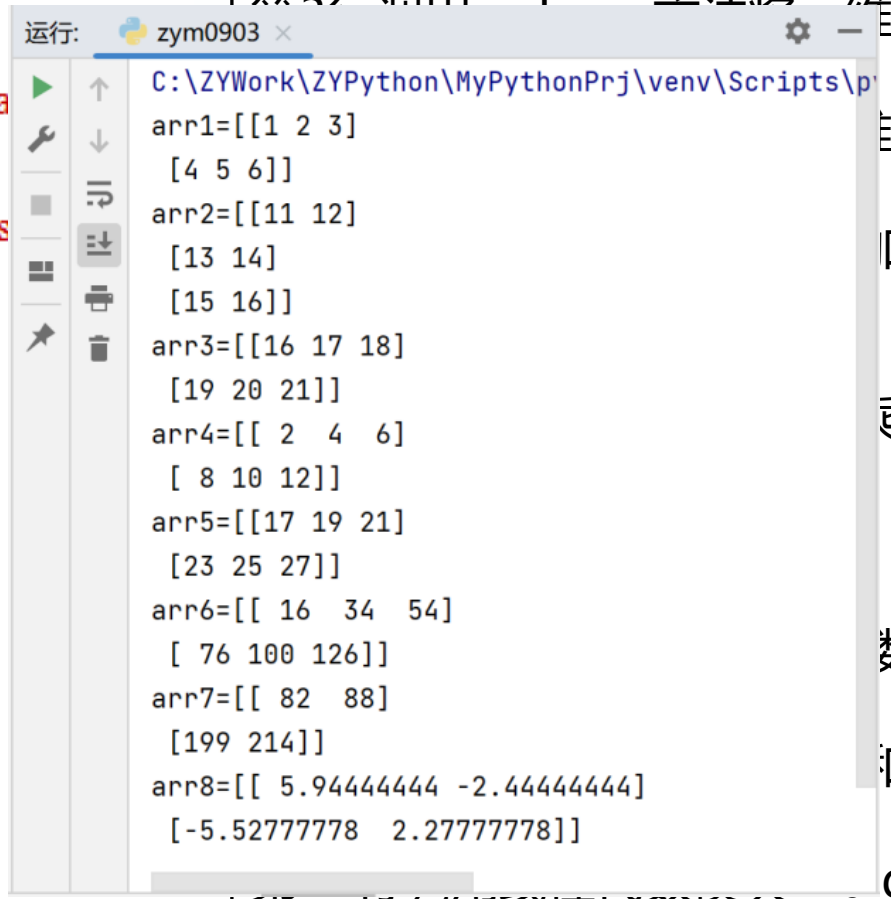
（5）两个矩阵的矩阵积。要求参与相乘运算的前一个矩阵的列数等于后一个矩阵的行数，此时，前一个矩阵的第i行的元素与后一个矩阵的第j列的元素的乘积取和，作为结果矩阵的第i行第j列的元素。

（6）矩阵求逆。求一个矩阵的逆矩阵，使之与原矩阵的矩阵积为单位矩阵。

9.1.3 矩阵运算

下面的实例展示了上述矩阵运算：

```
1   import numpy as np
2   if __name__=='__main__':
3       arr1=np.arange(1,6+1).resha
4       print(f'arr1={arr1}')
5       arr2=np.arange(11,16+1).res
6       print(f'arr2={arr2}')
7       arr3=15+arr1
8       print(f'arr3={arr3}')
9       arr4=2*arr1
10      print(f'arr4={arr4}')
11      arr5=arr1+arr3
12      print(f'arr5={arr5}')
13      arr6=arr1*arr3
14      print(f'arr6={arr6}')
15      arr7=np.dot(arr1,arr2)
16      print(f'arr7={arr7}')
17      arr8=np.linalg.inv(arr7)
18      print(f'arr8={arr8}')
```



数组转化为2行3列的矩阵arr1。

数组转化为3行2列的矩阵arr2。

加，即将15与矩阵arr1的每个元素

乘以2，赋给arr4。

，得到的矩阵赋给arr5。

数量积，结果赋给arr6。

和arr2的矩阵积，结果赋给arr7。

g中的inv方法计算arr7的逆矩阵，结果赋给arr8。

第18行输出arr8。

9.1.4 常用方法

程序包numpy常用的数值计算方法如下表所示。在表中，np表示numpy的别名；设矩阵a为“[[1 2 3 4] [5 6 7 8] [9 10 11 12]]”，即矩阵a由语句“a=np.arange(1, 12+1).reshape(3,4)”得到；设矩阵b为“[[1.1 2.2 3.3][4.4 5.5 6.6]]”，即矩阵b由语句“b=np.array([[1.1,2.2,3.3],[4.4,5.5,6.6]])”得到；设矩阵d为“d=np.array([[7,1,6,9],[11,19,8,2]])”。

9.1.4 常用方法

序号	方法名	含义及示例
1	floor	向下取整，例如：np.florr(b)得到[[1. 2. 3.][4. 5. 6.]]
2	ceil	向上取整，例如：np.ceil(b)得到[[2. 3. 4.][5. 6. 7.]]
3	around	四舍五入取整，例如：np.aroun
4	sin	求正弦（以弧度为单位），例如 [[0.01745241 0.0348995 0. [0.08715574 0.10452846 0. [0.15643447 0.17364818 0.1 这里，np.pi 为
5	cos	求余弦值（以弧
6	tan	求正切值（以弧
7	arcsin	求反正弦函数值（
8	arcos	求反余弦函数值（
9	arctan	求反正切函数值（
10	degrees	由弧度值转化为度，例如：np.degr np.degrees(np.pi/
11	sort	按升序排序，例如：np.sort(d,axis=1,k 7 9] [2 8 11 19]]”，这里，“axis=1 示按列排序；“kind='quicksort'”表示 的排序方法，此还，还有 stable 和 m （一种混合排序方法）或基数排序方法。
12	argsort	按升序排序后数组元素的原索引号构成的数组。例如：np.argsort(d,axis=1)将得到“[[1 2 0 3] [3 2 0 1]]”
13	mean	求数组的平均值。例如：np.mean(a)将得到 6.5；np.mean(a,axis=1)将得到“[2.5 6.5 10.5]”（axis=1 表示按行求平均值，axis=0 表示按列求平均值）
14	min	求最小值。例如，np.min(a)返回 1；np.min(a,axis=1)返回“[1 5 9]”。
15	max	求最大值。例如，np.max(a)返回 12；np.max(a,axis=1)返回“[4 8 12]”。
16	sum	求数组元素的和。例如，np.sum(a)将得到 78；np.sum(a,axis=1)将得到“[10 26 42]”。
17	median	求中位数。例如，np.median(a)将得到 6.5；np.median(a,axis=1)将得到“[2.5 6.5 10.5]”
18	percentile	求百分位数。例如，np.percentile(a,30)将得到 30%分位数 4.3；np.percentile(a,30,axis=1)将得到“[1.9 5.9 9.9]”。
19	var	求方差。例如，np.var(a,axis=1)将得到 11.916666666666666；np.var(a,axis=1)将得到“[1.25 1.25 1.25]”。
20	std	求标准差（或均方差）。例如，np.std(a)将得到 3.452052529534663；np.std(a,axis=1)将得到“[1.11803399 1.11803399 1.11803399]”。

以上内容仅为本文档的试下载部分，为可阅读页数的一半内容。如要下载或阅读全文，请访问：
https://d.book118.com/418116050006006101