- Numpy包

```
矩阵：
Xm = np.mean(X,axis=0)  #axis=0 上下压扁，axis=1左右压扁
print("Xm.shape",Xm.shape)    #Xm.shape (10,)
X1 = X - Xm[None,:]   #None表示自适应，即每行都用这一个行向量
print("X1.shape",X1.shape)    #X1.shape (442, 10)
# Compute the correlations per features
syy = np.mean(y1**2)
Sxx = np.mean(X1**2,axis=0)
Sxy = np.mean(X1*y1[:,None],axis=0)

# Compute the coefficients and R^2 value per feature
beta1 = Sxy/Sxx
beta0 = ym - beta1*Xm
Rsq = Sxy**2/Sxx/syy   #矩阵对应位置的数相乘或除
print(Rsq.shape)    #(10,)

#剔除长度为1的轴：直接改变原数组
np.squeeze(logreg.coef_)
np.squeeze() #可以直接进行压缩维度，官方doc的说明如下

>>> x = np.array([[[0], [1], [2]]])
>>> x.shape
(1, 3, 1)
>>> np.squeeze(x).shape
(3,)
>>> np.squeeze(x, axis=(2,)).shape
(1, 3)
#ravel flattern   也是降到1维数组
a1=b.ravel()   #改变a1影响b
a2=b.flattern()  #改变a1不影响b

#reshape:第一行然后第二行然后第三行；-1表示自适应；生成新数据，不改变a
d = a.reshape((-1,1))
d=np.reshape(a, (2, 3))


#矩阵合并
A = np.column_stack((np.ones(n,), X))

#生成网格:X和Y同维度，X按x成行扩展，Y按y成列扩展
X, Y = np.meshgrid(x,y)


```

```
#生成随机数
#（2，4）是维度，两行四列
#标准正态分布中随机生成，期望0标准差1
arr1 = np.random.randn(2,4)
#[0,1）中随机生成
arr2 = np.random.rand(2,4)
#[0,1）中随机生成
np.random.uniform(0,1,(nsamp,nx))
#随机打乱矩阵
numpy.random.shuffle(x) #直接打乱原矩阵
numpy.random.permutation(x)  #返回一个打乱的副本

```

```
#生成数组
numpy.linspace(0,1,10)  #start,end,num 左右都闭 endpoint改变是否包含end
numpy.arange(0,1,0.1)  #start,end stepsize  左闭右开
#改变数据类型
y0=np.linspace(0,0,10).astype(int)
y0.dtype  #查看数据类型
```

一、常用属性和函数

.ndim： 维度
.shape： 各维度的尺度 （2，5）
.size： 元素的个数 10
.dtype： 元素的类型 dtype('int32')
.itemsize： 每个元素的大小，以字节为单位，每个元素占4个字节
ndarray数组的创建
np.arange(n)；元素从0到n-1的ndarray类型
np.ones(shape)：生成全1
np.zeros((shape)， ddtype = np.int32)： 生成int32型的全0
np.full(shape, val)：生成全为val
np.eye(n)：生成单位矩阵>>> a = np.ones((2,2))

矩阵合并

x1 = np.array([0,0,1,1])

x1.shape=(1,4)

```
array([[0, 0, 1, 1]])
```

把行向量变成列向量！！

(4,)代表行向量

>>> b = np.eye(2)

>>> print np.vstack((a,b))

[[1. 1.]

 [1. 1.]

 [1. 0.]

 [0. 1.]]

>>> print np.hstack((a,b)) 注意这里要加两个括号！里面是个tuple

[[1. 1. 1. 0.]

 [1. 1. 0. 1.]]

最小二乘拟合，返回第一行是参数

```
out = np.linalg.lstsq(A,y_tr)
beta = out[0]
```

np.ones_like(a)：按数组a的形状生成全1的数组
np.zeros_like(a)：同理
np.full_like (a, val)：同理

np.linspace（1,10,4）： 根据起止数据等间距地生成数组
np.linspace（1,10,4, endpoint = False）： endpoint 表示10是否作为生成的元素
np.concatenate():

- 数组的维度变换

.reshape(shape)：不改变当前数组，依shape生成
.resize(shape)：改变当前数组，依shape生成
.swapaxes(ax1, ax2)：将两个维度调换
.flatten()：对数组进行降维，返回折叠后的一位数组

- 数组的类型变换

数据类型的转换： a.astype(new_type)：eg, a.astype (np.float)
数组向列表的转换： a.tolist()
数组的索引和切片

- 一维数组切片

a = np.array ([9, 8, 7, 6, 5,])
a[1:4:2] –> array([8, 6])： a[起始编号：终止编号（不含）： 步长]

- 多维数组索引

a = np.arange(24).reshape((2, 3, 4))
a[1, 2, 3] 表示 3个维度上的编号， 各个维度的编号用逗号分隔

- 多维数组切片

a [：，：，：：2] 缺省时，表示从第0个元素开始，到最后一个元素
数组的运算

imax = np.argmax(Rsq) 返回数组最大值的index

np.abs(a) np.fabs(a)：取各元素的绝对值
np.sqrt(a)：计算各元素的平方根
np.square(a)：计算各元素的平方
np.log(a) np.log10(a) np.log2(a)：计算各元素的自然对数、10、2为底的对数
np.ceil(a) np.floor(a)：计算各元素的ceiling 值， floor值（ceiling向上取整，floor向下取整）
np.rint(a)：各元素 四舍五入
np.modf(a)：将数组各元素的小数和整数部分以两个独立数组形式返回
np.exp(a)：计算各元素的指数值
np.sign(a)：计算各元素的符号值 1（+），0，-1（-）
.
np.maximum(a, b) np.fmax()：比较（或者计算）元素级的最大值
np.minimum(a, b) np.fmin()：取最小值
np.mod(a, b)：元素级的模运算
np.copysign(a, b)：将b中各元素的符号赋值给数组a的对应元素

- numpy随机数函数

numpy 的random子库 x = np.random.rand(10) 范围是（0,1]

numpy.random.uniform(low,high,size) 均匀分布中随机采样

rand(d0, d1, …,dn)：各元素是[0, 1）的浮点数，服从均匀分布
randn(d0, d1, …,dn)：标准正态分布
randint(low， high,（ shape）): 依shape创建随机整数或整数数组，范围是[low, high）
seed(s)： 随机数种子

shuffle(a)：根据数组a的第一轴进行随机排列，改变数组a
permutation(a)：根据数组a的第一轴进行随机排列， 但是不改变原数组，将生成新数组
choice(a[, size, replace, p])：从一维数组a中以概率p抽取元素， 形成size形状新数组，replace表示是否可以重用元素，默认为False。

- numpy的统计函数

sum(a, axis = None)：依给定轴axis计算数组a相关元素之和，axis为整数或者元组
mean(a, axis = None)：同理，计算平均值
average(a, axis =None, weights=None)：依给定轴axis计算数组a相关元素的加权平均值
std（a, axis = None） ：同理，计算标准差
var（a, axis = None）：计算方差
eg： np.mean(a, axis =1)： 对数组a的第二维度的数据进行求平均
a = np.arange(15).reshape(3, 5)
np.average(a, axis =0, weights =[10, 5, 1])：对a第一各维度加权求平均，weights中为权重，注意要和a的第一维匹配

min(a) max(a)：计算数组a的最小值和最大值
argmin(a) argmax(a)：计算数组a的最小、最大值的下标（注：是一维的下标）
unravel_index(index, shape)：根据shape将一维下标index转成多维下标
ptp(a)：计算数组a最大值和最小值的差
median(a)：计算数组a中元素的中位数（中值）
eg：a = [[15, 14, 13],
[12, 11, 10]]
np.argmax(a) –> 0
np.unravel_index(np.argmax(a), a.shape) –> (0,0)

*numpy提供的一些常量

```
>>> np.Inf #无穷大
inf
>>> np.NAN #非数字
nan
>>> np.Infinity #无穷大
inf
>>> np.MAXDIMS #数组的最大维度
32
>>> np.NINF #负无穷大
```

```
-inf
>>> np.NaN #非数字
nan
>>> np.NZERO #负0
-0.0
>>> np.pi
```

常用数学函数

NumPy提供常见的如sin,cos和exp
更多函数all, alltrue, any, apply along axis, argmax, argmin, argsort, average, bincount, ceil, clip, conj, conjugate, corrcoef, cov, cross, cumprod, cumsum, diff, dot, floor, inner, inv, lexsort, max, maximum, mean, median, min, minimum, nonzero, outer, prod, re, round, sometrue, sort, std, sum, trace, transpose, var, vdot, vectorize

数组的subset

```
x = np.random.rand(10)  # 10 random elements from 0 to 1
x1 = x[2:5]             # Elements 2,3,4  (Note 5 is NOT included)
x2 = x[:4]              # Elements 0,1,2,3  (Starts at 0, element 4 NOT included)
x3 = x[7:]              # Elements 7,8,9 (Element 7 IS included.  Ends at 9 NOT 10)
xlast = x[-1]           # The last element
```

二、Array数组生成方法

Type code C Type Minimum size in bytes
'c' character 1
'b' signed integer 1
'B' unsigned integer 1
'u' Unicode character 2
'h' signed integer 2
'H' unsigned integer 2
'i' signed integer 2
'I' unsigned integer 2
'l' signed integer 4
'L' unsigned integer 4
'f' floating point 4
'd' floating point 8

```
from array import *
myarray=array("l") //表示创建一个interger类型的数组
myarray.append(3)//追加元素
myarray.pop()//删除最后一个
myarray.remove(X)//删除指定的一个x
num=myarray[0]//取第一个值
myarray.insert(3,10)//3表示下标
myarray.reverse()//数组反序
```

Numpy:
NumPy数组是一个多维数组对象，称为ndarray。其由两部分组成:

实际的数据
描述这些数据的元数据
大部分操作仅针对于元数据，而不改变底层实际的数据。

关于**NumPy**数组有几点必需了解的:

NumPy数组的下标从0开始。
同一个NumPy数组中所有元素的类型必须是相同的。

在详细介绍NumPy数组之前。先详细介绍下NumPy数组的基本属性。NumPy数组的维数称为秩（rank），一维数组的秩为1，二维数组的秩为2，以此类推。在NumPy中，每一个线性的数组称为是一个轴（axes），秩其实是描述轴的数量。比如说，二维数组相当于是两个一维数组，其中第一个一维数组中每个元素又是一个一维数组。所以一维数组就是NumPy中的轴（axes），第一个轴相当于是底层数组，第二个轴是底层数组里的数组。而轴的数量——秩，就是数组的维数。

NumPy的数组中比较重要ndarray对象属性有：

1. ndarray.ndim：数组的维数（即数组轴的个数），等于秩。最常见的为二维数组（矩阵）。
2. ndarray.shape：数组的维度。为一个表示数组在每个维度上大小的整数元组。例如二维数组中，表示数组的“行数”和“列数”。ndarray.shape返回一个元组，这个元组的长度就是维度的数目，即ndim属性。
3. ndarray.size：数组元素的总个数，等于shape属性中元组元素的乘积。
4. ndarray.dtype：表示数组中元素类型的对象，可使用标准的Python类型创建或指定dtype。另外也可使用前一篇文章中介绍的NumPy提供的数据类型。
5. ndarray.itemsize：数组中每个元素的字节大小。例如，一个元素类型为float64的数组itemsiz属性值为8(float64占用64个bits，每个字节长度为8，所以64/8，占用8个字节），又如，一个元素类型为complex32的数组item属性为4（32/8）。
6. ndarray.data：包含实际数组元素的缓冲区，由于一般通过数组的索引获取元素，所以通常不需要使用这个属性。

代码：

```
from numpy import *

a=array([[1,2],[2,3]])
a.ndim//数组的维数
2
a.shape//数组的维度，即行数和列数
(2L, 2L)
a.dtype//数组中元素的类型
dtype('int32')
a.size//数组元素的总个数
4
```

注意：

使用array函数创建时，参数必须是由方括号括起来的列表，而不能使用多个数值作为参数调用array。

- >>> a = array(1,2,3,4)　# 错误
- >>> a = array([1,2,3,4]) # 正确

可以在创建时显式指定数组中元素的类型：

- >>> c = array([[1,2], [3,4]], dtype=complex)
- >>> c
- array([[1.+0.j, 2.+0.j],
- [3.+0.j, 4.+0.j]])

NumPy提供一个类似arange的函数返回一个数列形式的数组:
arange(4)
输出—-array([0, 1, 2, 3])
arange(10,30,5)//10-30步长为5
输出—-array([10, 15, 20, 25])

```
x = np.array(range(1,5))**2
```

当arange使用浮点数参数时，由于浮点数精度有限，通常无法预测获得的元素个数。因此，最好使用函数linspace去接收我们想要的元素个数来代替用range来指定步长。
np.linspace(10,30,5)//这里5表示的是要输出5个数
输出—array([10., 15., 20., 25., 30.])

保证右边界也输出的方法：

```
step = 2.5
x2 = np.arange(0,10+step/2,step)
print(x2)
```

操作可以矢量化（对数组中每个元素进行操作）
Most operations can be vectorized meaning that the operation is applied to each component.

In [16]:
```
x = np.array(range(2,6))
y = x**2
print("x = " + str(x))
print("y = " + str(y))
```