Отчет Кобака Ф.А. 18ДКК-1
по вычислительной части в лекции о многомерном шкалировании

Вычисления проводил с использованием ЯП Python3.

Входные данные

14650	1223	10.8	148.4	0.264
4364.3	606	0.2	76.5	0.234
1400	306	8.2	157.3	0.274

Для нормализации данных по формуле $z_{ij}=x_{ij}/\bar{x}_j$ функция:

```
def normalize_data(data):
'''funciton for normalisation given data'''
#inputs:
#<data> - data as np array
# outputs:
#<norm_data> - normalised data

    result = copy(data)

    for i in range(data.shape[0]):
        for j in range(data.shape[1]):
            result[i,j] = data[i,j]/np.mean(data[:,j]);
     return result;
```

После применения функции будут получены нормализованные данные:

2,1529026221811	1,7185011709601	1,6875	1,1648351648351	1,0259067357513
0,6413592432755	0,8515222482435	0,03125	0,6004709576138	0,9093264248704
0,2057381345429	0,4299765807962	1,28125	1,2346938775510	1,0647668393782

Функция для вычисления попарных расстояний между объектами:

```
def eucl_distance(p1, p2):
        return(np.sum(np.square(p1-p2))**(1/2))

def object_distances(data, dist_fun = eucl_distance):
'''fucniton for computing pair distances'''
#inputs:
#<data> - data about objects where objects in lines
#<dist_fun> - funciton for computing distances wich takes two numpy.array with lines
# the Euclidean distance is selected by default
#outputs - numpy.array wich contains pair distances

        n = data.shape[0];

        result = np.zeros(shape = (n,n))

        for i in range(n):
                for j in range(i, n):
                result[i,j] = dist_fun(data[i,:], data[j,:])
```

```
        result[j,i] = result[i,j]

    return result
```

Результат применения — **матрица попарных расстояний:**

0	2,47218078394906	2,37132337973422
2,47218078394906	0	1,53504623212236
2,37132337973422	1,53504623212236	0

Немного округлим числа для того, чтобы сходилось с примером из конспекта.

0	2,47	2,38
2,47	0	1,54
2,38	1,54	0

Функция реализующая формулу для получения матрицы B:

```
def b_matrix(comp_matr):
'''function realises matrix transformation from matrix pairs comparison to b matrix'''
# inputs:
# <comp_matr> - simetric matrix with comparison comp_matr[i, j] = comp_matr[j, i]
characterize the similarity of object(i) and object(j)
# outputs - transformed matrix needed for required coordinates

    result = copy(comp_matr);

    n = comp_matr.shape[0];
    # sum of squares matrix elements
    mat_ss = np.square(comp_matr).sum();


    for i in range(n):
        for j in range(i, n):
            line_sq_sum = np.square(comp_matr[i,:]).sum();
            col_sq_sum = np.square(comp_matr[:,j]).sum();
            result[i,j] = -(1/2)*(comp_matr[i,j]**2 - ((1/n)*line_sq_sum +
            (1/n)*col_sq_sum) + (1/(n**2))*mat_ss);
            result[j,i] = result[i,j];

    return result;
```

после применения будет получена матрица B:

2,351	-1,24825	-1,10275
-1,24825	1,2534	-0,00515
-1,10275	-0,00515	1,1079

Далее функция для проведения метода главных компонент, позволяющая получить все промежуточные вычисления:

```
def principal_comp_full_compution(X):
	z = data_stand(X)

	R = np.corrcoef(X, rowvar = False)
	# данная функция также создана мной(ее можно найти тут)
	# https://github.com/Dranikf/multivariate_statistical_analysis
	[L, V] = eig_matlab(R)

	L_sqrt = np.sqrt(abs(L))

	A = np.dot(V, L_sqrt)
	inv_A = np.linalg.inv(A)

	return {'F' : np.dot(inv_A, z.transpose()).transpose(), 'A': A, 'L' : L,
	'z': z, 'V' : V, 'R':R, 'L_sqrt': L_sqrt}
```

Полученная корреляционная матрица R

1	-0,881650349134506	-0,845470899902699
-0,881650349134506	1	0,493403243655527
-0,845470899902699	0,493403243655527	1

Собсвенные числа корреляционной матрицы :
2,49276137191224
0
0,507238628087761

Собсвенные векторы (по столбцам) V:

0,633272178501146	-0,773523005736087	0,02507005252134
-0,552987922357975	-0,474910996535083	-0,684590317705608
-0,54145240387556	-0,419668565617041	0,728496801208016

корни из собственных чисел $\Lambda^{1/2}$:

1,57884811552988	0	0
0	0	0
0	0	0,712206871693724

Матрица факторных нагрузок $A = V\Lambda^{1/2}$

0,999840585644033	0	0,017855063679421
-0,873083939125669	0	-0,487569928564924
-0,85487110750805	0	0,518840427827246

Как видно она не совпадает с приведенной в лекционном материале