PYTHON COURSE FOR GEOSCIENTIST

BATCH VII

MODUL

SEG SC UPer

@segsc.up

SEG SC UNIVERSITAS PERTAMINA

SEGSC UNIVERSITAS PERTAMINA

A. Basic Data Transformation

```
## bivariate

import numpy as np
import matplotlib.pyplot as plt
from scipy import interpolate

data = np.loadtxt('F02-1_logs.las', skiprows=37)
data = data[6000:8000,:]

depth = data[:, 0]
rhob = data[:, 1]
dt = data[:, 2]
vp = 1/dt*(10**6)

# data normalization
s_rhob  = np.std(rhob, ddof=1)
mean = np.mean(rhob)

print(s_rhob)
print(mean)

rhob_n = (rhob - mean)/s_rhob

s_rhob_n  = np.std(rhob_n, ddof=1)
mean_n = np.mean(rhob_n)

print(s_rhob_n)
print(mean_n)

plt.subplot(1,2,1)
plt.plot(rhob, depth)
plt.ylim([np.max(depth), np.min(depth)])
plt.subplot(1,2,2)
plt.plot(rhob_n, depth)
plt.ylim([np.max(depth), np.min(depth)])
plt.show()

## normal score transform
rhobsort = np.sort(rhob)
freq = np.linspace(0,1,len(rhob))

f = interpolate.interp1d(rhobsort, freq)
rhob_norm = f(rhob)

z = np.linspace(-3,3,101)
```

@segsc.up
SEG SC UNIVERSITAS PERTAMINA
seg.up.sc@gmail.com

```
pdf = 1/(2*np.pi)*np.exp(-0.5*z**2)
cdf = np.cumsum(pdf)
cdf = (cdf - np.min(cdf))/(np.max(cdf) - np.min(cdf))
f2 = interpolate.interp1d(cdf, z)
rhob_nst = f2(rhob_norm)

plt.subplot(1,2,1)
plt.plot(rhob, depth)
plt.ylim([np.max(depth), np.min(depth)])
plt.subplot(1,2,2)
plt.plot(rhob_nst, depth)
plt.ylim([np.max(depth), np.min(depth)])
plt.show()

bin = int(np.sqrt(len(rhob)))

plt.subplot(1,2,1)
plt.hist(rhob, bin)
plt.subplot(1,2,2)
plt.hist(rhob_nst, bin)
plt.show()
```

B. Estimation of Empty Log

```
## bivariate

import numpy as np
import matplotlib.pyplot as plt
from scipy import stats

data1 = np.loadtxt('F02-1_logs.las', skiprows=35)

data = data1[data1[:, 1] != -999.2500, :]

depth = data[:, 0]
rhob = data[:, 1]/1000
dt = data[:, 2]
vp = 1/dt*(10**6)
GR = data[:, 3]

r = np.zeros((4,))
rhob_n = (rhob - np.mean(rhob))/np.std(rhob,ddof=1)

r[0] = stats.pearsonr(dt, rhob)[0]
r[1] = stats.pearsonr(dt, np.sqrt(rhob))[0]
r[2] = stats.pearsonr(dt, np.log(rhob))[0]
r[3] = stats.pearsonr(dt, rhob_n)[0]
```

```
prew = 0.001
K = np.vstack((rhob, np.log(rhob), np.sqrt(rhob), rhob_n, np.ones_like(dt))).T

m = np.linalg.inv(K.T@K + np.eye(K.shape[1])*prew)@K.T@dt
# m = np.linalg.pinv(K)@dt

dt_pred = K@m
vp_pred = 1/dt_pred*(10**6)     # Predicted Vp

bad = np.sign(np.diff(vp))      # find the bad-interpolated value of Vp
n = np.where(bad == 0)[0]       # find the bad-interpolated value of Vp

vp_new = vp*1
vp_new[n] = vp_pred[n]          # corrected Vp

plt.subplot(1,2,1)
plt.plot(vp, depth)
plt.ylim(np.max(depth), np.min(depth))
plt.subplot(1,2,2)
plt.plot(vp_new, depth)
plt.ylim(np.max(depth), np.min(depth))
plt.show()
```

C. Principal Component Analysis

Principal Component is the method to reduce the dimensionality of our data/variables. The key of PCA is eigenvector.

Assumptions :

- **Not Biased**
- **Linearly Independent**: one attribute is not resulted from linear combination of other attributes.
- **Normally distributed**

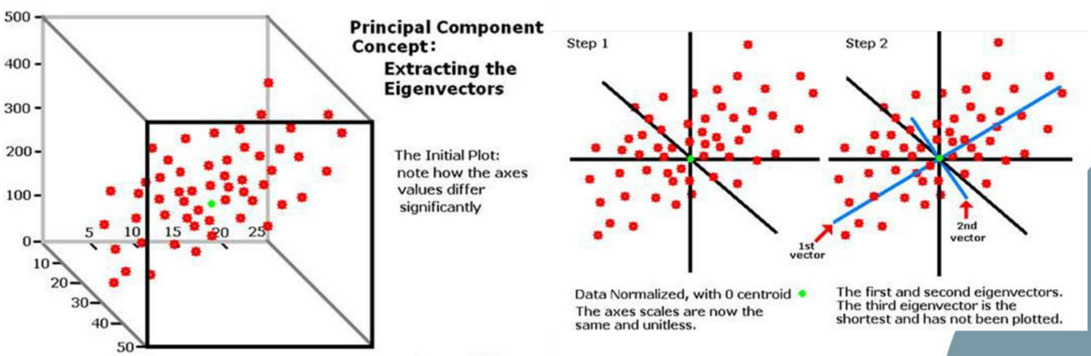

Fig 1. Concept of PCA (GeoSoftware's Help)

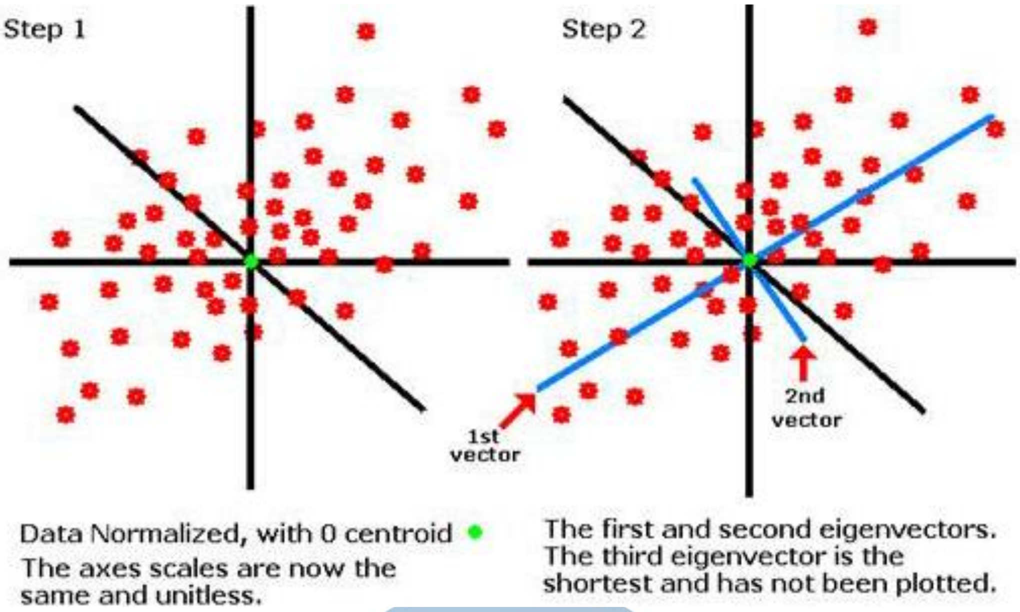

Fig 2. Principal and Subsequent Eigenvectors (GeoSoftware's Help).

- **First Principal Component**: PC which is derived from eigenvector with highest eigenvalue
- **Other Principal Components**: PC which are derived from eigenvectors with second-highest to smallest eigenvalues

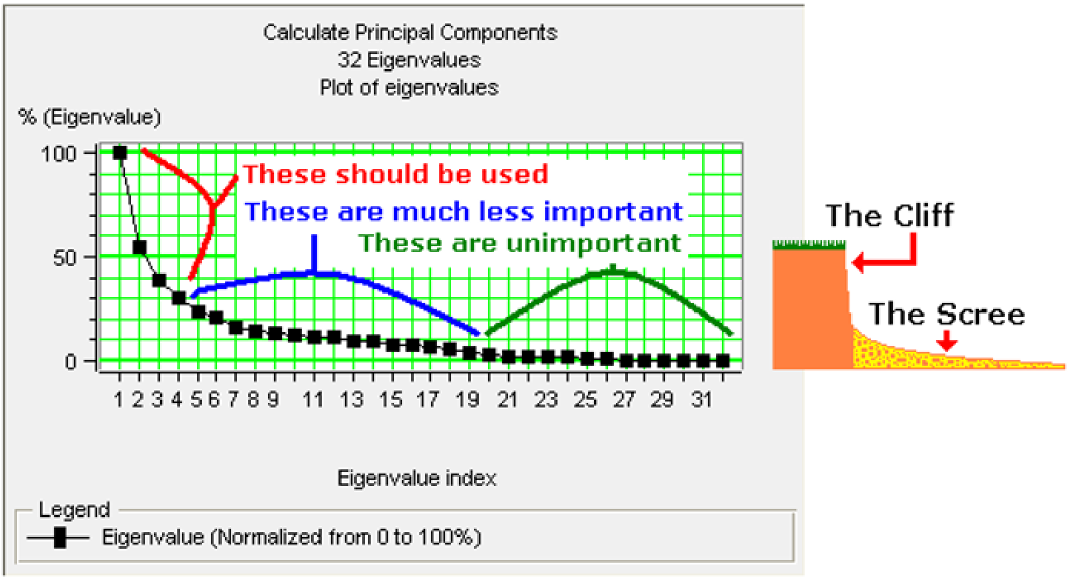

Fig 3. Eigenvalues (GeoSoftware's Help).

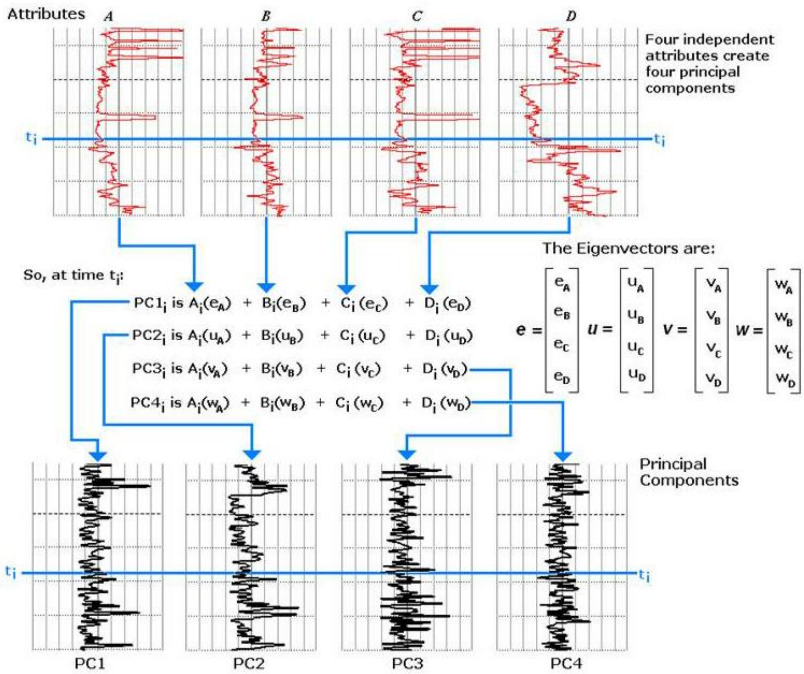

Fig 4. Transformation original inputs to PCA (GeoSoftware's Help).