# ENSZ adatok klaszterezése

## 1. feladat: adatbetöltés

Töltsük be a /volumes/data/clustering/UN.csv fájl következő oszlopait: 'country','region','lifeMale', 'lifeFemale', 'infantMortality', 'GDPperCapita'

A feladat során csak ezekkel az oszlopokkal fogunk dolgozni!

In [ ]:
```
import pandas as pd

DATA_PATH = '/volumes/data/clustering/'

def load_un_data(filename):
    pass # TODO!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!


df = load_un_data(DATA_PATH + 'UN.csv')
df.head()
```

## 2.feladat: NaN-t tartalmazó cellák és sorok száma

Számoljuk meg, hogy összesen hány NaN érték van a táblában, majd azt is adjuk meg, hogy hány sor tartalmaz legalább egy NaN értéket!

In [ ]:
```
print "Number of cells with NaN: %d" % 0 # it should be 38 TODO!!!!!!!!!!!!!!!!
!
print "Number of rows with NaN: %d" % 0 # it should be 19 TODO!!!!!!!!!!!!!!!!!!
```

## 3. feladat: Távolítsuk el a NaN-t tartalmazó sorokat

In [ ]:
```
pass # TODO!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

In [ ]:
```
df.head()
```

## 4. feladat: Elemezzük a kapott táblázatot scatter plot és hisztogramok segítségével

Mit mondhatunk a következőkről?

- Élethossz eloszlása a különböző régiókban
- Élethossz és a GDP kapcsolata
- Gyermekhalálozás (infant mortality) és a GDP kapcsolata régiónként
- Nők és férfiak élethossza közötti különbség régiónként? Hogyan korrelál ez a GDP-vel?

Melyik kérdést milyen típusú ábrával tudjuk megválaszolni?

In [ ]:
```
import matplotlib.pyplot as plt
%matplotlib inline

def list_of_regions(df):
    pass # TODO!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

def life_in_region(df, region):
    pass # TODO!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

for region in list_of_regions(df):
    plt.figure(figsize=(16,2))
    plt.xlim((0,100))
    plt.xlabel('Age')
    plt.ylabel('P')
    life_in_region(df, region).hist(bins=range(0,100,5), label=region, alpha=0.5
, normed=True)
    plt.legend()
    plt.show()
```

In [ ]:
```
def life_vs_gdp(df):
    pass # TODO!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

lvgdp = life_vs_gdp(df)
regions = pd.factorize(df['region'].append(df['region']))
plt.figure(figsize=(16,16))
plt.xlabel('Age')
plt.ylabel('GDP')
plt.yscale('log')
plt.scatter(....) # TODO!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
plt.show()
```

In [ ]:
```
def mortality_vs_GDP(df, region):
    pass # TODO!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

plt.figure(figsize=(18,12))
n=1
for region in list_of_regions(df):
    mvgdp = mortality_vs_GDP(df,region)
    plt.subplot(2,3,n)
    n+=1
    plt.xlim((0,200))
    plt.xlabel('infMortality')
    plt.ylabel('GDP')
    plt.yscale('log')
    plt.scatter(x=mvgdp['infantMortality'].values, y=mvgdp['GDPperCapita'].value
s, s=20, alpha=0.5, label=region)
    plt.legend()
plt.show()
```

In [ ]:
```
def difference_life(df, region):
    pass # TODO!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

def gdp_region(df, region):
    pass # TODO!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

plt.figure(figsize=(18,6))
n=1
for region in list_of_regions(df):
    plt.subplot(2,3,n)
    n+=1
    plt.xlabel('Difference')
    plt.ylabel('P')
    plt.xlim((-10,10))
    difference_life(df, region).hist(bins=range(-10,10,1), label=region, normed=
True)
    plt.legend()
plt.show()

plt.figure(figsize=(18,12))
n=1
for region in list_of_regions(df):
    plt.subplot(2,3,n)
    n+=1
    plt.xlabel('Difference')
    plt.ylabel('GDP')
    plt.xlim((-10,10))
    plt.ylim((0,50000))
    plt.scatter(x=difference_life(df,region), y=gdp_region(df, region), s=20, al
pha=0.5, label=region)
    plt.legend()
plt.show()
```

## 5. feladat: Klaszterezzük K-Means segítségével az adatpontokat

Végezzük el az adatok klaszterezését a következő változók figyelembe vételével:

- 'lifeMale', 'lifeFemale', 'infantMortality', 'GDPperCapita'

Nem tudjuk, hogy hány klaszterre van szükségünk. Probáljuk ki a K=1,2,3,4,5,6,7,8,9,10 eseteket. Mindegyikre számoljuk ki az Inertia és a Silhouette Score értéket és ábrázoljuk K függvényében egy grafikonon. Melyik K értéket érdemes választani?

In [ ]:
```
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import silhouette_score

def generate_models(K_array, train):
    pass # TODO!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

# NOTE: Use 100 sample points to calculate the silhouette score
def get_silhouette_scores(train, model_array):
    pass # TODO!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

K_array = range(1,11,1)
scaler = StandardScaler()
train = scaler.fit_transform(df[['lifeMale', 'lifeFemale', 'infantMortality', 'G
DPperCapita']].values)
#train = df[['lifeMale', 'lifeFemale', 'infantMortality', 'GDPperCapita']].value
s # ALSO TRY THIS WITHOUT SCALER...

models = generate_models(K_array, train)

plt.figure(figsize=(18,3))
plt.plot(K_array, [m.inertia_ for m in models], marker='o', markersize=6,
      markeredgewidth=2, markeredgecolor='r', markerfacecolor='None')
plt.grid(True)
plt.xlabel('Number of clusters')
plt.ylabel('Inertia: Average within-cluster sum of squares')
plt.show()

plt.figure(figsize=(18,3))
plt.plot(...) # TODO!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
plt.grid(True)
plt.xlabel('Number of clusters')
plt.ylabel('Silhouette score')
plt.show()
```

## 6.feladat: A klaszterek vizualizációja

K=3 még mindkét mértékben jó klaszterezést mutat. Válasszuk ezt! Osztályozzuk a pontokat, majd különböző változó párok mentén ábrázoljuk azokat scatter ploton!

In [ ]:
```
bestmodel = models[2]

clusters = bestmodel.predict(train)

# Visualize lifeMale/GDP
# TODO

# Visualize lifeFemale/GDP
# TODO

# Visualize infantMortality/GDP
# TODO
```

## 7.feladat: Mit történik, ha nem használunk StandardScaler-t a 'train' mátrix előállításánál

A K-Means és a legtöbb klaszterező esetén a távolság fogalmának jelentős szerepe van a pontok csoportosításában. A változók értékei eltérő skálán mozoghatnak, így kiemelten fontos őket összeskálázni a modellezés előtt. Hasonló igaz pl. az SVM osztályozóra is.

## 8.feladat: Listázzuk ki az egyes klaszterekbe került régiókat és országokat

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: