In [3]:
```python
# For plotting
from matplotlib import pyplot as plt
import seaborn as sns
%matplotlib inline
import numpy as np
```

In [4]:
```python
import warnings
warnings.filterwarnings('ignore')
```

# PCA, and Regression on PCA output

use Google/StartPage/Ecosia/Bing/Duckduckgo for each of the unknown terms, and read on them, you will know a lot

In [5]:
```python
# data exploration
import pandas as pd
df = pd.read_csv('./data-for-code/no-empty-cell-mortality_subgroup_data_june_9th_gender_neutral-based_data_after
df.head()
```

Out[5]:

| | Age-group: From USRDS | Age-group: To USRDS | Gender | Actual Dark-green vegetables Intake | Actual Red and orange vegetables Intake | Actual Starchy vegetables Intake | Actual Other vegetables Intake | Actual Whole grains intakes | Actual Taken Refined grains amount | Avg Meat, Poultry and Eggs subgroup taken | ... | Avg Solid Fats taken | Avg Milks and milk drinks taken | Avg Water, noncarbonated intake | Avg Alcoholic beverages intake |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 4 | Neutral | 53.14 | 59.44 | 68.81 | 94.21 | 171.86 | 56.76 | 109.77 | ... | 29.13 | 478.86 | 360.73 | 360.00 |
| 1 | 5 | 9 | Neutral | 77.63 | 59.86 | 75.65 | 106.01 | 270.11 | 95.13 | 166.41 | ... | 38.78 | 376.15 | 588.17 | 797.02 |

8 rows × 23 columns

[7]:
```python
df_actual_only = df.drop(['Age-group: From USRDS', 'A
df_actual_only.head()
```

9]:
```python
y = abs(standardisedX['ESRD patients: Avg. Annual Mortality rates']) > 0.5
standardisedX_with_target = standardisedX
standardisedX = standardisedX.drop(['ESRD patients: Avg. Annual Mortality rates', 'ESRD p
                                    'Dialysis patients: Total (or %) deaths for target yea
```


9]:
```python
y = abs(standardisedX['ESRD patients: Avg. Annual Mortality rates']) > 0.5
standardisedX_with_target = standardisedX
standardisedX = standardisedX.drop(['ESRD patients: Avg. Annual Mortality rates', 'ESRD p
                                    'Dialysis patients: Total (or %) deaths for target yea
```

9]:
```
y = abs(standardisedX['ESRD patients: Avg. Annual Mortality rates']) > 0.5
standardisedX_with_target = standardisedX
standardisedX = standardisedX.drop(['ESRD patients: Avg. Annual Mortality rates', 'ESRD p
                                    'Dialysis patients: Total (or %) deaths for target yea
```

10]:
```
from sklearn import decomposition
#pca = decomposition.PCA(n_components=2).fit(standardisec
pca = decomposition.PCA().fit(standardisedX)
pca
```

10]:
```
PCA(copy=True, iterated_power='auto', n_components=None,
  svd_solver='auto', tol=0.0, whiten=False)
```

]:
```
#ref: https://python-for-multivariate-analysis.readthedocs.io/a_little_book_of_python_for_mu
def pca_summary(pca, standardised_data, out=True):
    names = ["PC"+str(i) for i in range(1, len(pca.explained_variance_ratio_)+1)]
    a = list(np.std(pca.transform(standardised_data), axis=0))
    b = list(pca.explained_variance_ratio_)
    c = [np.sum(pca.explained_variance_ratio_[:i]) for i in range(1, len(pca.explained_varia
    columns = pd.MultiIndex.from_tuples([("sdev", "Standard deviation"), ("varprop", "Propor
    summary = pd.DataFrame( list(zip(a, b, c)), index=names, columns=columns)
    if out:
        print("Importance of components:")
        display(summary)
    return summary
```

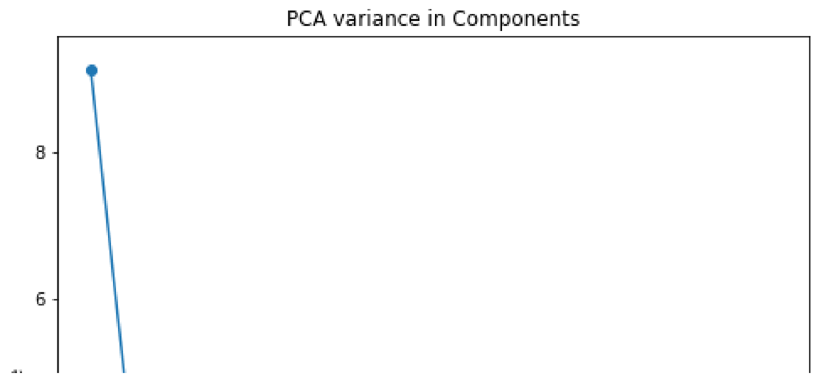
]:
```
####----
summary = pca_summary(pca, standardisedX)
####----
```

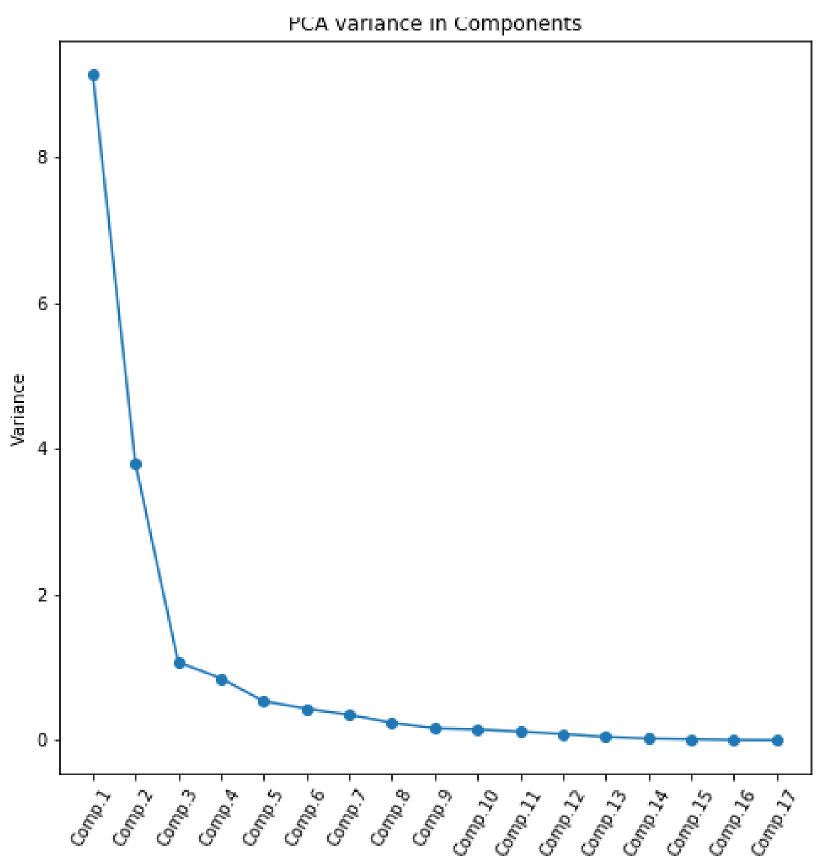
```
Importance of components:
```

| | sdev | varprop | cumprop |
|---|---|---|---|
| | Standard deviation | Proportion of Variance | Cumulative Proportion |
| **PC1** | 3.021925 | 5.371782e-01 | 0.537178 |
| **PC2** | 1.948069 | 2.232338e-01 | 0.760412 |
| **PC3** | 1.034469 | 6.294863e-02 | 0.823361 |
| **PC4** | 0.921359 | 4.993539e-02 | 0.873296 |

# Important Components

```
def screeplot(pca, standardised_values):
    y = np.std(pca.transform(standardised_values), axis=0)**2
    x = np.arange(len(y)) + 1
    plt.plot(x, y, "o-")
    plt.xticks(x, ["Comp."+str(i) for i in x], rotation=60)
    plt.ylabel("Variance")
    plt.title('PCA variance in Components')
    plt.savefig('../../progress_reports/to_submit/pca_univariate_bivar:
    plt.show()

screeplot(pca, standardisedX)
#plt.savefig('../../progress_reports/to_submit/pca_univariate_bivariat
```



```
16]: # comp 3 to 4 has the most slope change - comp 7 starts the flat line
     # first three are the most important
     # Will retain first five though first three will be analyzed primarily
```

```
17]: #summary.sdev**2
     #pca.components_[0]
     #np.sum(pca.components_[0]**2)
```

```
18]: ####----
     # ref: https://python-for-multivariate-analysis.readthedocs.io/a_little_book
     # not my code, I am using this as (similar to) a library function with some
     def calcpc(variables, loadings):
         # find the number of samples in the data set and the number of variables
         numsamples, numvariables = variables.shape
         # make a vector to store the component
         pc = np.zeros(numsamples)
         # calculate the value of the component for each sample
         for i in range(numsamples):
             valuei = 0
             for j in range(numvariables):
                 valueij = variables.iloc[i, j]
                 loadingj = loadings[j]
                 valuei = valuei + (valueij * loadingj)
             pc[i] = valuei
         return pc
```

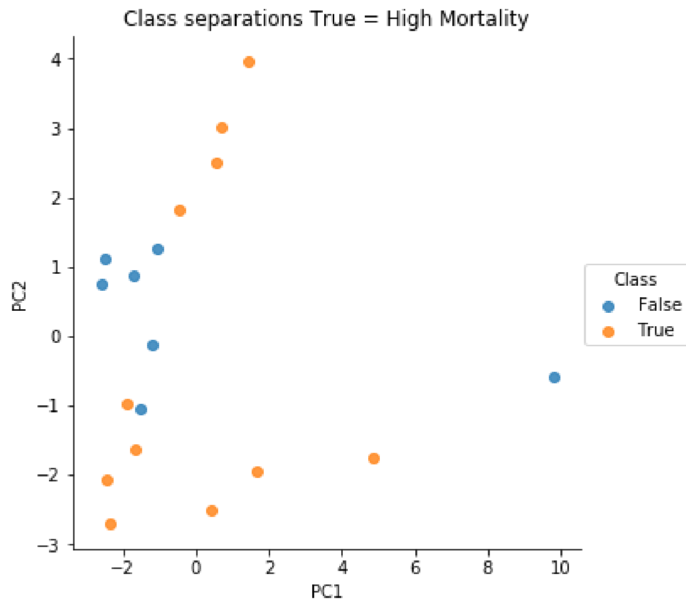
```
In [19]: ####----
         calcpc(standardisedX, pca.components_
         pca.transform(standardisedX)[:, 0]
         pca.transform(standardisedX)[:, 0]
         pca.components_[1]
         np.sum(pca.components_[1]**2)
         #highest loadings for

Out[19]: 0.9999999999999989
```

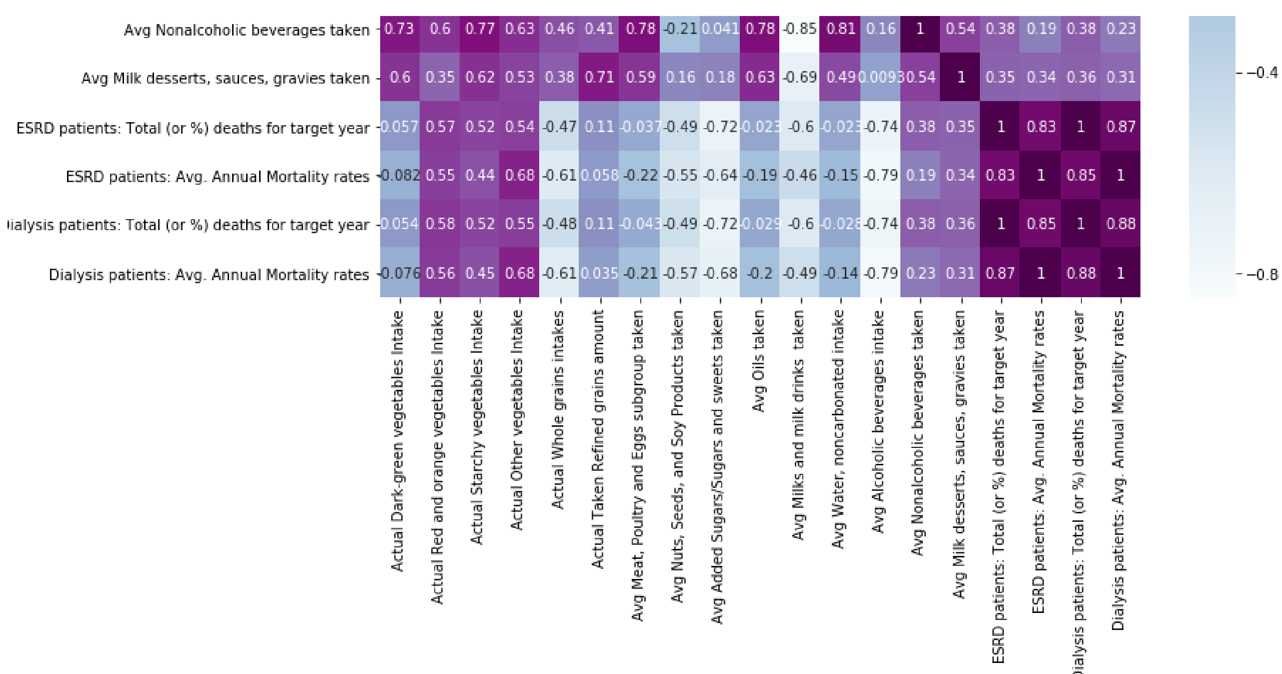
```
[21]: ####----
# # ref: https://python-for-multivariate-analysis.readthedoc
# not my code from the URL above, using this as a library fu
def pca_scatter(pca, standardised_values, classifs):
    foo = pca.transform(standardised_values)
    bar = pd.DataFrame(list(zip(foo[:, 0], foo[:, 1], classi
    #plt.savefig('../../progress_reports/to_submit/pca_univa
    sns.lmplot("PC1", "PC2", bar, hue="Class", fit_reg=False

pca_scatter(pca, standardisedX, y)

# y can be used as classes like High, low, neutral mortality

# plt.suptitle('Mortality class and Principle components, y
plt.title('Class separations True = High Mortality')
plt.savefig('../../progress_reports/to_submit/pca_univariate
standardisedX
```

```
standardisedX
```

t[21]:

| | Actual Dark-green vegetables Intake | Actual Red and orange vegetables Intake | Actual Starchy vegetables Intake | Actual Other vegetables Intake | Actual Whole grains intakes | Actual Taken Refined grains amount | Avg Meat, Poultry and Eggs subgroup taken | Avg Seafood taken | F |
|---|---|---|---|---|---|---|---|---|---|
| 0 | -2.680917 | -1.244130 | -2.343451 | -1.787702 | -2.314888 | -3.719039 | -2.839852 | -2.471056 | -( |
| 1 | -1.253327 | -1.214110 | -2.033392 | -1.496430 | -0.649816 | -0.086621 | -1.455389 | -1.917166 | ( |
| 2 | -0.538075 | -0.892465 | -1.115000 | -1.511487 | 0.155011 | 0.919701 | -0.687139 | 0.151259 | ( |
| 3 | -0.838282 | -1.274865 | -0.624526 | -0.968191 | 0.424643 | 1.274707 | -0.145967 | -0.555477 | 1 |
| 4 | 1.139591 | 0.256879 | 0.454336 | -0.225941 | 0.827141 | -0.341279 | 0.530866 | 1.466881 | 3 |
| 5 | 1.426392 | -1.276294 | 0.048629 | 0.065577 | 1.180662 | 0.061061 | 1.099903 | 0.082335 | ( |


Class separations True = High Mortality
PC2
PC1
Class
False
True


Avg Nonalcoholic beverages taken
Avg Milk desserts, sauces, gravies taken
ESRD patients: Total (or %) deaths for target year
ESRD patients: Avg. Annual Mortality rates
Dialysis patients: Total (or %) deaths for target year
Dialysis patients: Avg. Annual Mortality rates
Actual Dark-green vegetables Intake
Actual Red and orange vegetables Intake
Actual Starchy vegetables Intake
Actual Other vegetables Intake
Actual Whole grains intakes
Actual Taken Refined grains amount
Avg Meat, Poultry and Eggs subgroup taken
Avg Nuts, Seeds, and Soy Products taken
Avg Added Sugars/Sugars and sweets taken
Avg Oils taken
Avg Milks and milk drinks taken
Avg Water, noncarbonated intake
Avg Alcoholic beverages intake
Avg Nonalcoholic beverages taken
Avg Milk desserts, sauces, gravies taken
ESRD patients: Total (or %) deaths for target year
ESRD patients: Avg. Annual Mortality rates
Dialysis patients: Total (or %) deaths for target year
Dialysis patients: Avg. Annual Mortality rates