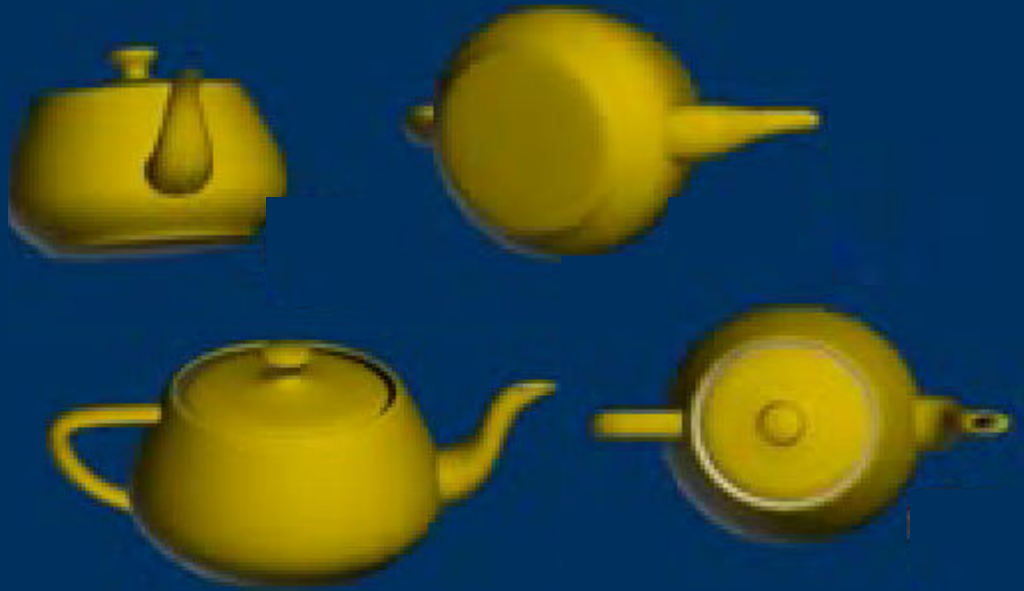
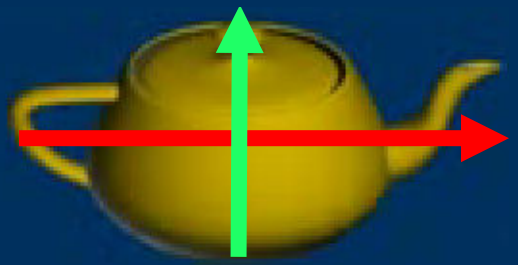
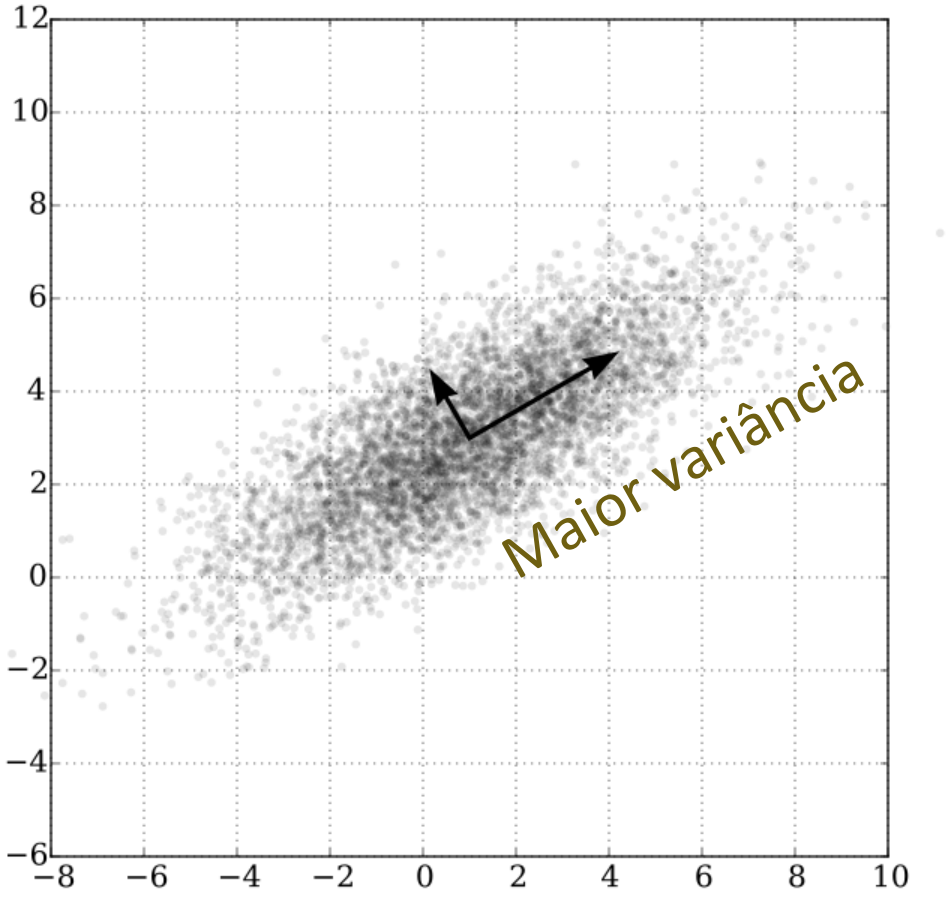
Maior quantidade de informação?

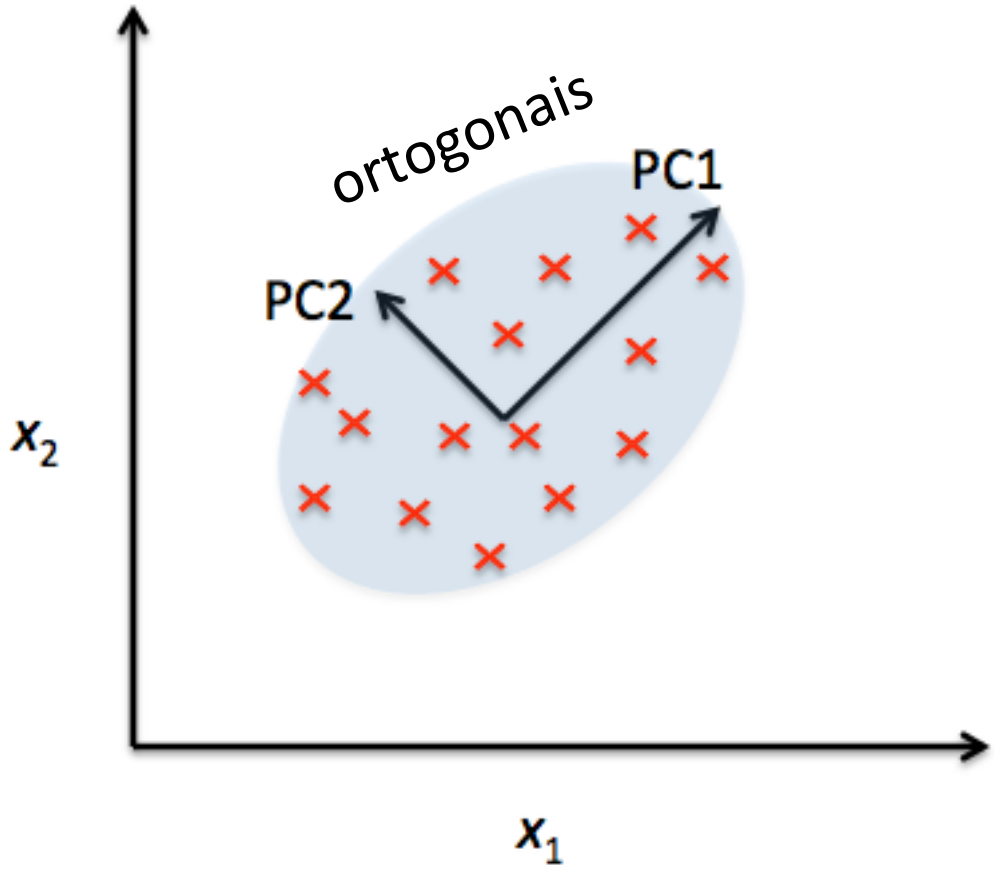


1ª componente principal
2ª componente principal



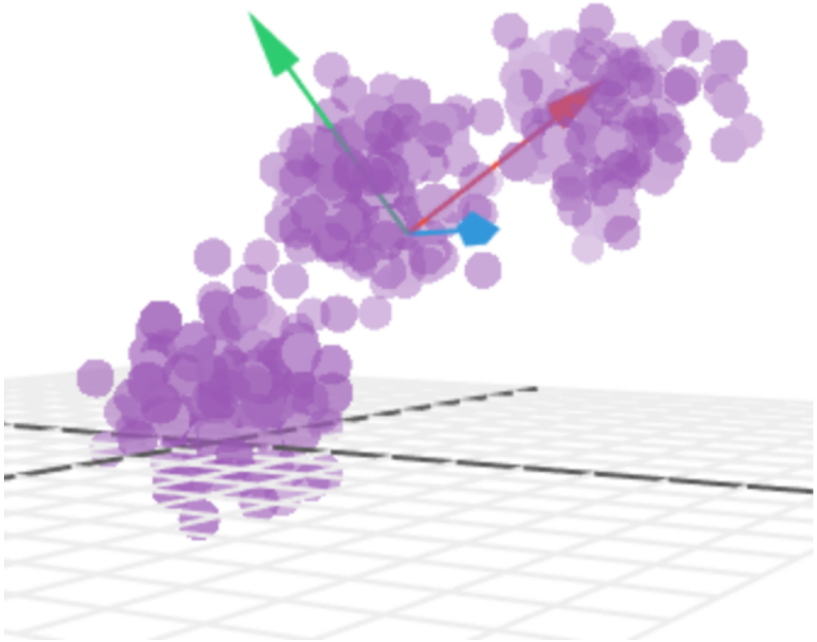
- Resumir dados com muitas variáveis independentes para um pequeno conjunto de variáveis derivadas da originais
  - 1ª componente = variância máxima
  - 2ª componente = 2ª maior variância
  - etc...
- Covariância entre quaisquer componentes obtidas deve ser zero


12
10
8
6
4
2
0
−2
−4
−6
−8
−6
−4
−2
0
2
4
6
8
10
Maior variância

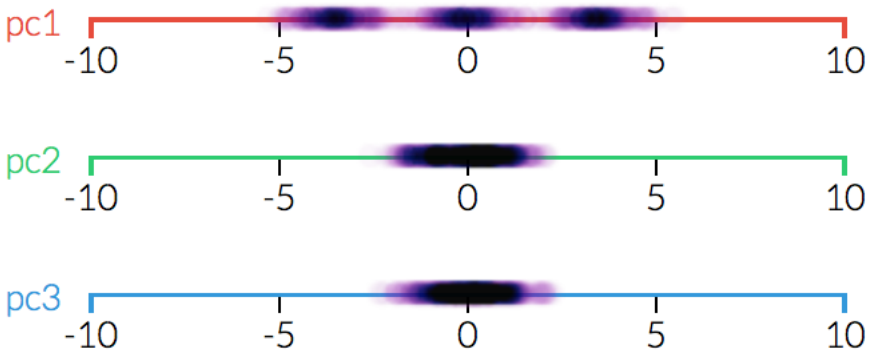

ortogonais
PC1
PC2
x_2
x_1

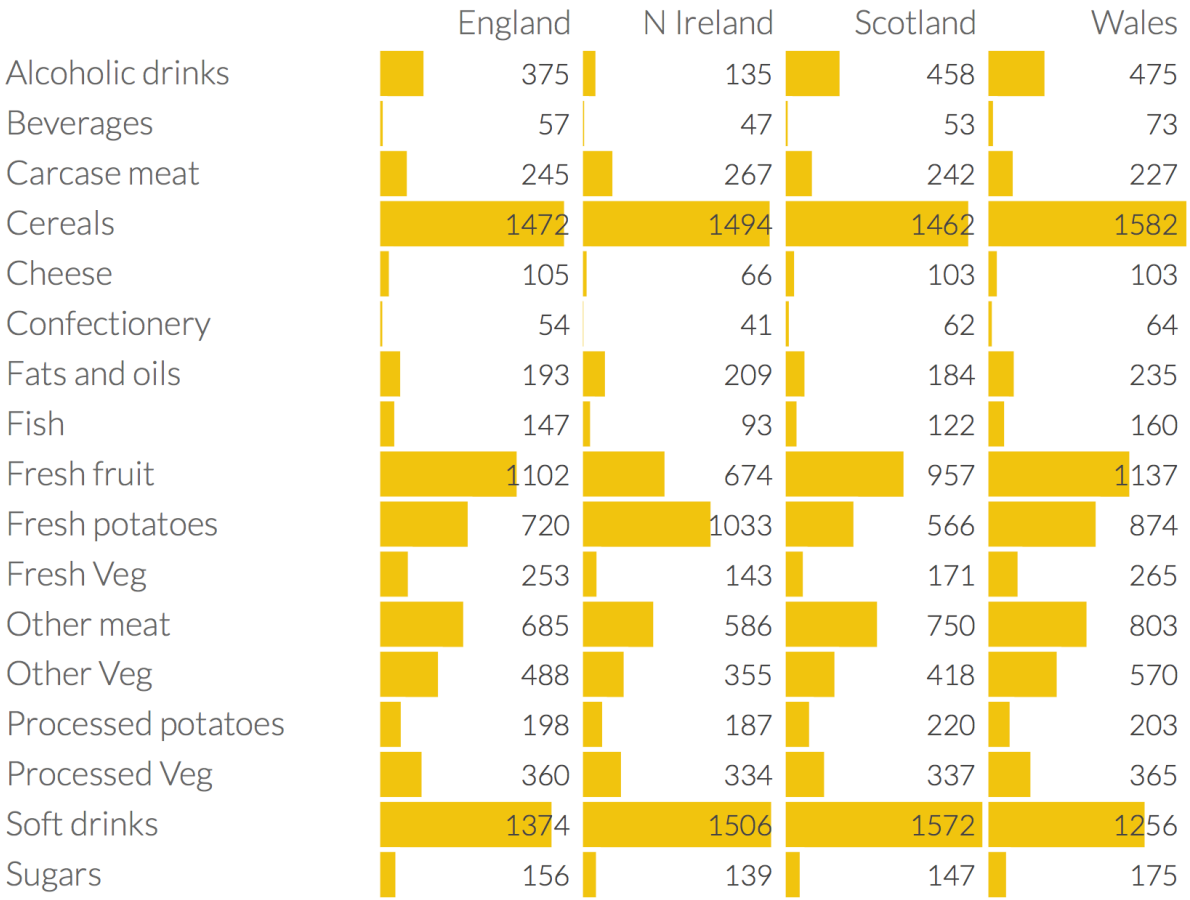
Principal Component Analysis (PCA)

Técnica de transformação linear não supervisionada que permite redução de dimensionalidade ao tempo que mantém as informações mais relevantes



Redução de dimensionalidade

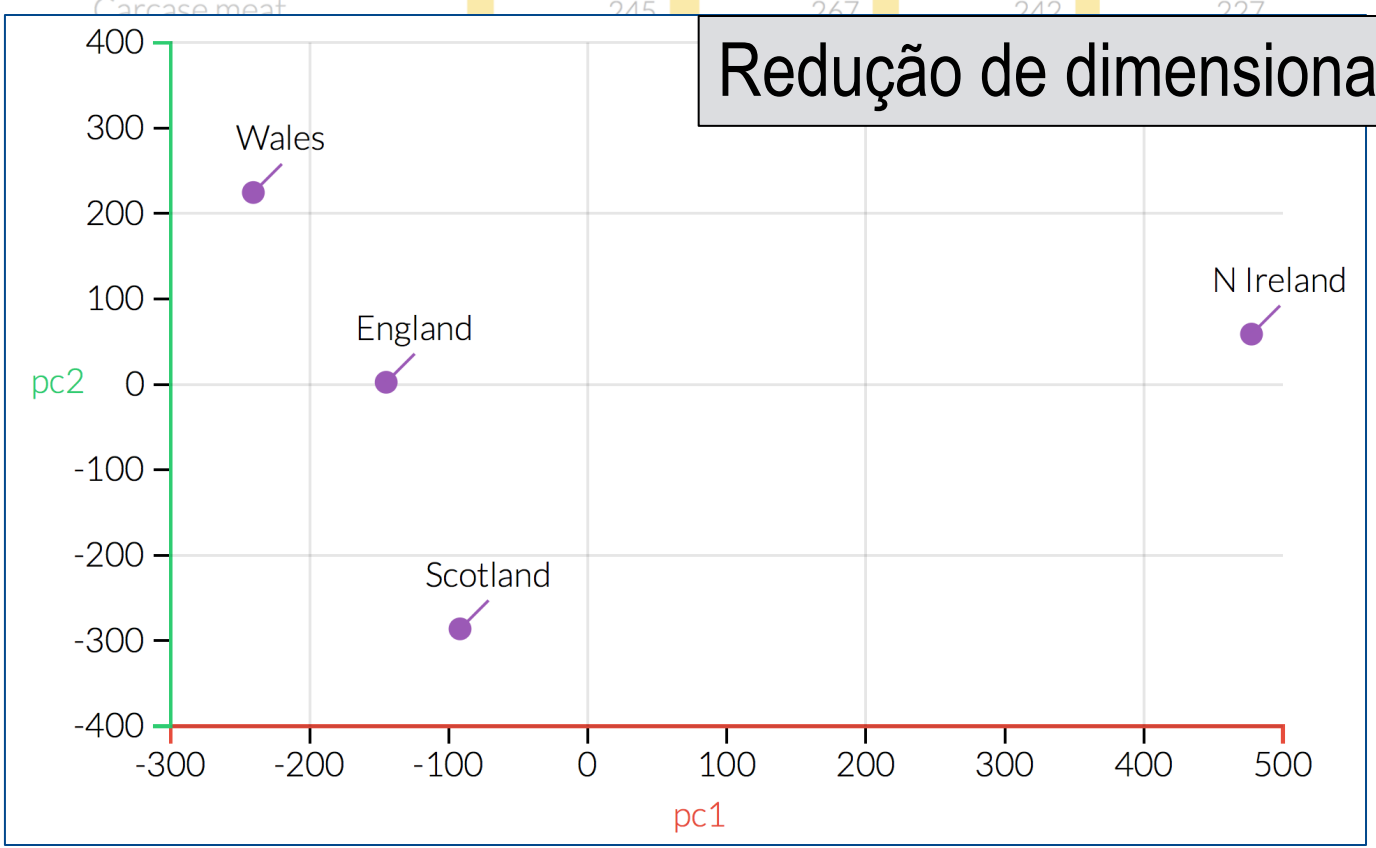




Consumo médio semanal de 17 tipos de comida (em gramas) per capita nos 4 países do Reino Unido
Tabela mostra variações entre diferentes tipos de comida, mas diferenças globais não são simples de se notar!

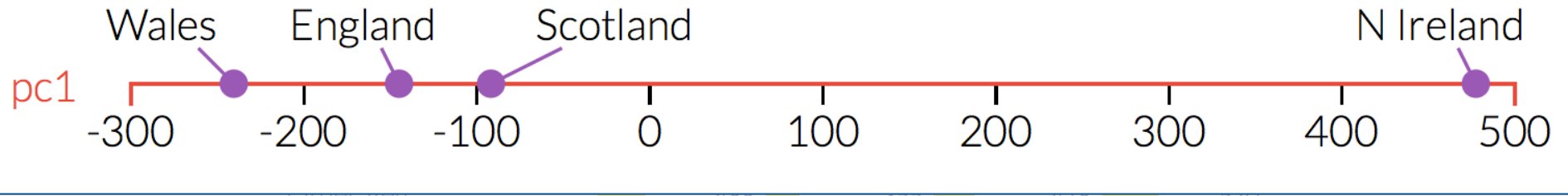
	England	N Ireland	Scotland	Wales
Alcoholic drinks	375	135	458	475
Beverages	57	47	53	73
Carcase meat	245	267	242	227

Redução de dimensionalidade

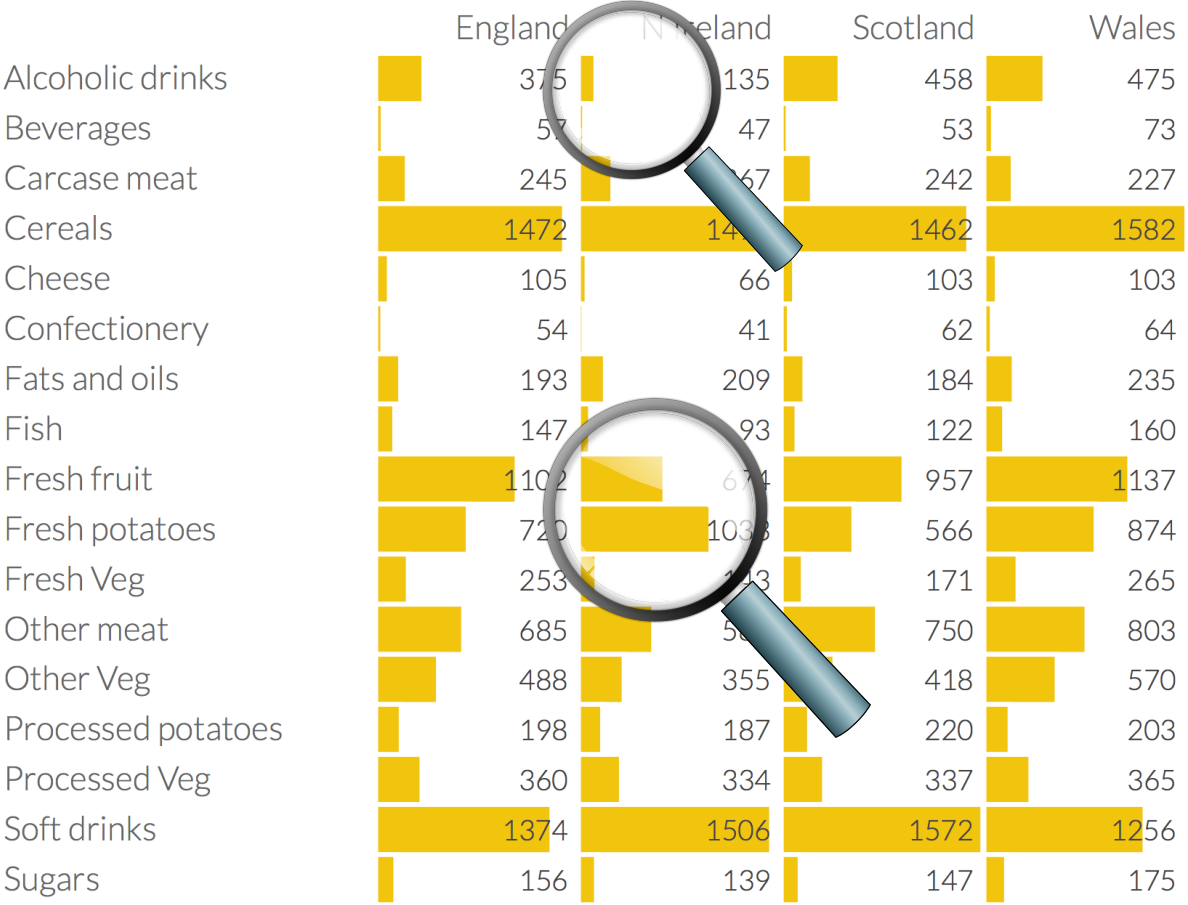


	England	N Ireland	Scotland	Wales
Alcoholic drinks	375	135	458	475
Beverages	57	47	53	73
Carcase meat	245	267	242	227
Cereals	1472			
Cheese	105			
Confectionery	54	41	62	64
Fats and oils	193	209	184	235

Redução de dimensionalidade



Processed potatoes	198	187	220	203
Processed Veg	360	334	337	365
Soft drinks	1374	1506	1572	1256
Sugars	156	139	147	175


England
N Ireland
Scotland
Wales
Alcoholic drinks 375 135 458 475
Beverages 57 47 53 73
Carcase meat 245 67 242 227
Cereals 1472 14 1462 1582
Cheese 105 66 103 103
Confectionery 54 41 62 64
Fats and oils 193 209 184 235
Fish 147 93 122 160
Fresh fruit 1102 957 1137
Fresh potatoes 720 566 874
Fresh Veg 253 171 265
Other meat 685 750 803
Other Veg 488 355 418 570
Processed potatoes 198 187 220 203
Processed Veg 360 334 337 365
Soft drinks 1374 1506 1572 1256
Sugars 156 139 147 175

A análise dos componentes permite atentar para um fato geográfico: a Irlanda do Norte é a única nação dentre as 4 que não se localiza na ilha da Grã Bretanha.

Redução de dimensionalidade

$$\mathbf{x} = [x_1, x_2, \ldots, x_j], \mathbf{x} \in \mathbb{R}^d$$

Matriz de transformação

$$\downarrow \mathbf{xW}, \quad \mathbf{W} \in \mathbb{R}^{d \times k}$$

$$\mathbf{z} = [z_1, z_2, \ldots, z_k], \quad \mathbf{z} \in \mathbb{R}^k$$

PCA .: Passos

1. Estandardizar o *dataset*; 
2. Construir a matriz de covariância;
3. Decompor a matriz de covariância em seus autovetores e autovalores;
4. Selecionar k autovetores que correspondem aos k maiores autovalores;
5. Construir matriz de projeção W a partir dos k autovetores;
6. Transformar o espaço de características de d para k usando W.

PCA .: 2. Matriz de co-variância

$$\Sigma = \begin{bmatrix} \sigma_1^2 & \sigma_{12} & \sigma_{13} \\ \sigma_{21} & \sigma_2^2 & \sigma_{23} \\ \sigma_{31} & \sigma_{32} & \sigma_3^2 \end{bmatrix}$$

$$\sigma_{jk} = \frac{1}{n} \sum_{i=1}^{n} \left(x_j^{(i)} - \mu_j \right) \left(x_k^{(i)} - \mu_k \right)$$

```
1 import numpy as np
2 cov_mat = np.cov(X_train_std.T)
3 eigen_vals, eigen_vecs = np.linalg.eig(cov_mat)
4
5 print('\nEigenvalues \n%s' % eigen_vals)
```

Σ

```
[[1.008 0.088 0.231 -0.329  0.214  0.356  0.299 -0.169  0.096 0.569 -0.048  0.074  0.633]
 [0.088 1.008 0.202  0.307 -0.060 -0.301 -0.412  0.366 -0.192 0.307 -0.550 -0.394 -0.203]
 [0.231 0.202 1.008  0.450  0.183  0.122  0.061  0.187 -0.026 0.246 -0.109 -0.018  0.159]
  ... ]
```

PCA .: passos

1. Estandardizar o *dataset*;
2. Construir a matriz de covariância;
3. Decompor a matriz de covariância em seus autovetores e autovalores;
4. Selecionar k autovetores que correspondem aos k maiores autovalores;
5. Construir matriz de projeção W a partir dos k autovetores;
6. Transformar o espaço de características de d para k usando W.

PCA .: 3. Autovetores e Autovalores

Principais componentes de $\sum$

Magnitude dos principais componentes

$$\Sigma\gamma = \lambda\gamma$$

Nota: Autovetores e Autovalores

$$\mathbf{A}\boldsymbol{\gamma} = \lambda\boldsymbol{\gamma} \tag{4a}$$

$$(\mathbf{A} - \lambda\mathbf{I})\,\boldsymbol{\gamma} = 0 \tag{4b}$$

$$\left|\mathbf{A} - \lambda\mathbf{I}\right| = 0 \tag{4c}$$

A equação (4c) é denominada *a equação característica* do operador T ou da matriz $\mathbf{A}$. As raízes desta equação são os autovalores de $\mathbf{A}$. Para calcular um autovetor $\boldsymbol{\gamma}$ associado ao autovalor λ, basta resolver a equação (4b).

Nota: Autovetores e Autovalores

Considere a transformação linear T de $\mathbb{R}^3 \to \mathbb{R}^3$ representada pela matriz $\mathbf{A}$.

$$T(x, y, z) = (3x - y + z, -x + 5y - z, x - y + 3z)$$

$$\mathbf{A} = \begin{bmatrix} 3 & -1 & 1 \\ -1 & 5 & -1 \\ 1 & -1 & 3 \end{bmatrix}$$

Nota: Autovetores e Autovalores

A equação característica da matriz anterior é:

$$|\mathbf{A} - \lambda\mathbf{I}| = \begin{vmatrix} 3-\lambda & -1 & 1 \\ -1 & 5-\lambda & -1 \\ 1 & -1 & 3-\lambda \end{vmatrix} = 0$$

$$|\mathbf{A} - \lambda\mathbf{I}| = (3-\lambda)\begin{vmatrix} 5-\lambda & -1 \\ -1 & 3-\lambda \end{vmatrix} - (-1)\begin{vmatrix} -1 & -1 \\ 1 & 3-\lambda \end{vmatrix} + (+1)\begin{vmatrix} -1 & 5-\lambda \\ 1 & -1 \end{vmatrix} = 0$$

$$\lambda^3 - 11\lambda^2 + 36\lambda - 36 = 0$$

$$(\lambda - 2)(\lambda - 3)(\lambda - 6) = 0$$

Os autovalores de $\mathbf{A}$ são $\lambda_1 = 2, \lambda_2 = 3, \lambda_3 = 6$. Substituindo λ_i $(i = 1, 2, 3)$ na equação (4b) permite calcular os autovetores correspondentes:

- Para $\lambda_1 = 2$ temos

$$\begin{bmatrix} 1 & -1 & 1 \\ -1 & 3 & -1 \\ 1 & -1 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

  o que admite uma infinidade de soluções do tipo: $x = -z,\ y = 0$ (ex. $\boldsymbol{\gamma}_1 = [1, 0, -1]^\top$).

- Para $\lambda_2 = 3$ temos

$$\begin{bmatrix} 0 & -1 & 1 \\ -1 & 2 & -1 \\ 1 & -1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

  solução: $x = y = z$ (ex. $\boldsymbol{\gamma}_2 = [1, 1, 1]^\top$).

- Para $\lambda_3 = 6$ temos

$$\begin{bmatrix} -3 & -1 & 1 \\ -1 & -1 & -1 \\ 1 & -1 & -3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

  solução: $y = -2x,\ x = z$ (ex. $\boldsymbol{\gamma}_3 = [1, -2, 1]^\top$).

```python
import numpy as np
cov_mat = np.cov(X_train_std.T)
eigen_vals, eigen_vecs = np.linalg.eig(cov_mat)

print('\nEigenvalues \n%s' % eigen_vals)
```

λ

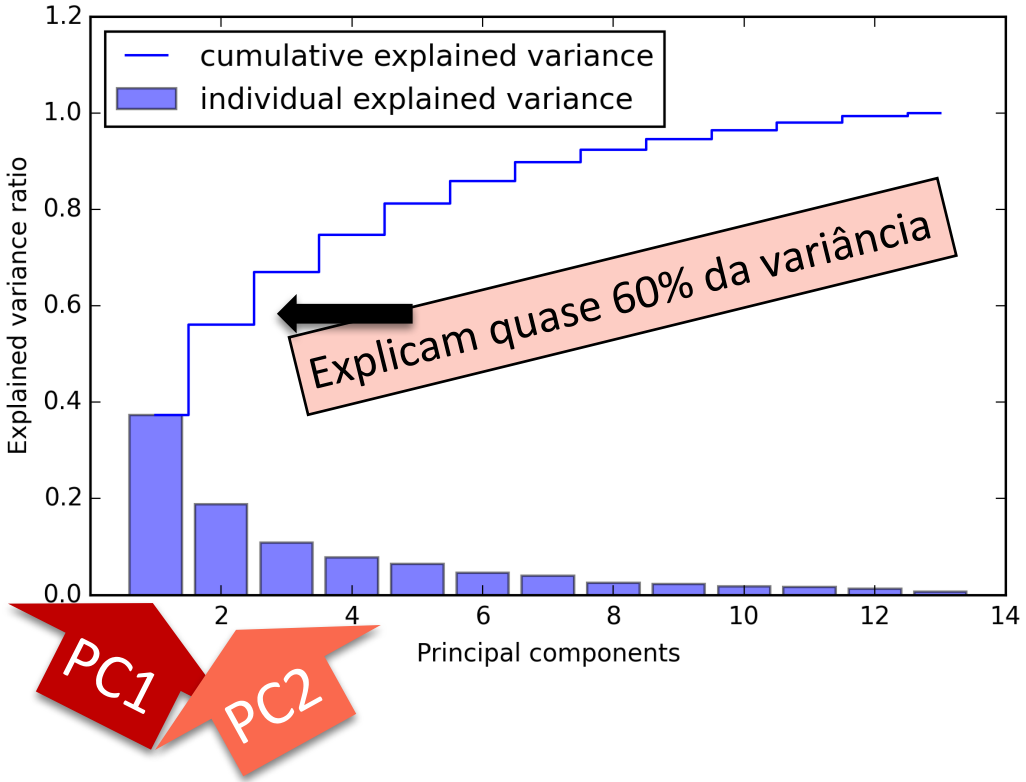
```
[4.892 2.466 1.428 1.012 0.849 0.602 0.523 0.084 0.331 0.296 0.168 0.214 0.240]
```

Matriz 13x13 → 13 autovalores e autovetores

Proporção da variância de autovalores

(Explained variance ratio)

$$\frac{\lambda_j}{\sum_{j=1}^{d} \lambda_j}$$



PCA .: passos

1. Estandardizar o *dataset*;
2. Construir a matriz de covariância;
3. Decompor a matriz de covariância em seus autovetores e autovalores;
4. Selecionar k autovetores que correspondem aos k maiores autovalores;
5. Construir matriz de projeção W a partir dos k autovetores;
6. Transformar o espaço de características de d para k usando W.

PCA .: 4. Selecionar k Autovetores

```
# Make a list of (eigenvalue, eigenvector) tuples
eigen_pairs = [(np.abs(eigen_vals[i]), eigen_vecs[:,i]) for i in range(len(eigen_vals))]

# Sort the (eigenvalue, eigenvector) tuples from high to low
eigen_pairs.sort(reverse=True)
```

PCA .: passos

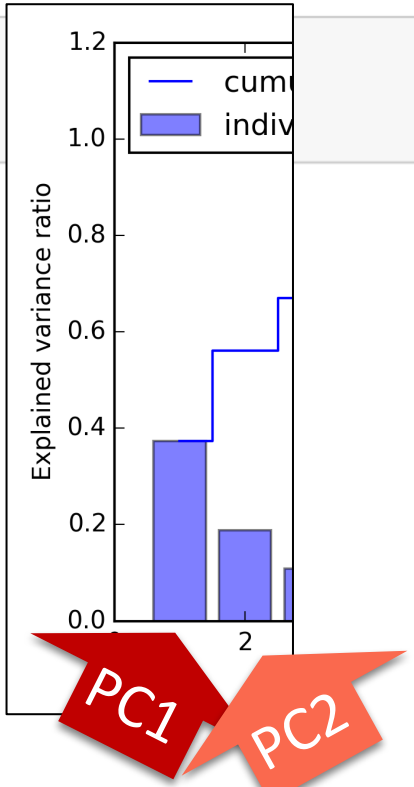
1. Estandardizar o *dataset*;
2. Construir a matriz de covariância;
3. Decompor a matriz de covariância em seus autovetores e autovalores;
4. Selecionar k autovetores que correspondem aos k maiores autovalores;
5. Construir matriz de projeção W a partir dos k autovetores;
6. Transformar o espaço de características de d para k usando W.

PCA .: 5. Matriz de projeção $\mathbf{W}$

... p/ k = 2

```
1 w = np.hstack((eigen_pairs[0][1][:, np.newaxis],
2                eigen_pairs[1][1][:, np.newaxis]))
3 print('Matrix W:\n', w)
```

```
Matrix W:
 [[ 0.14669811  0.50417079]
 [-0.24224554  0.24216889]
 [-0.02993442  0.28698484]
 [-0.25519002 -0.06468718]
 [ 0.12079772  0.22995385]
 [ 0.38934455  0.09363991]
 [ 0.42326486  0.01088622]
 [-0.30634956  0.01870216]
 [ 0.30572219  0.03040352]
 [-0.09869191  0.54527081]
 [ 0.30032535 -0.27924322]
 [ 0.36821154 -0.174365  ]
 [ 0.29259713  0.36315461]]
```



PCA .: passos

1. Estandardizar o *dataset*;
2. Construir a matriz de covariância;
3. Decompor a matriz de covariância em seus autovetores e autovalores;
4. Selecionar k autovetores que correspondem aos k maiores autovalores;
5. Construir matriz de projeção W a partir dos k autovetores;
6. Transformar o espaço de características de d para k usando W.

PCA .: 6. Transformação de características

1xk 1xd dxk

$$\mathbf{x}' = \mathbf{x}\mathbf{W}$$

1x13 13x2

```
1 X_train_std[0].dot(w)
```

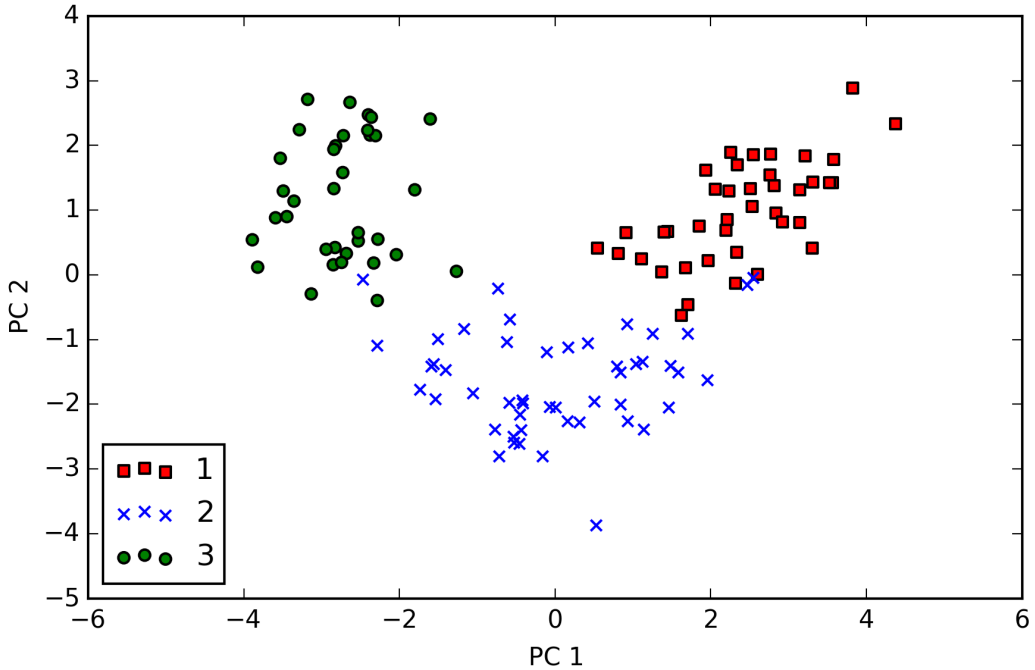
```
array([-2.59891628,  0.00484089])
```

1x2

```
X_train_pca = X_train_std.dot(w)
colors = ['r', 'b', 'g']
markers = ['s', 'x', 'o']

for l, c, m in zip(np.unique(y_train), colors, markers):
    plt.scatter(X_train_pca[y_train==l, 0],
                X_train_pca[y_train==l, 1],
                c=c, label=l, marker=m)

plt.xlabel('PC 1')
plt.ylabel('PC 2')
plt.legend(loc='lower left')
plt.tight_layout()
# plt.savefig('./figures/pca2.png
plt.show()
```



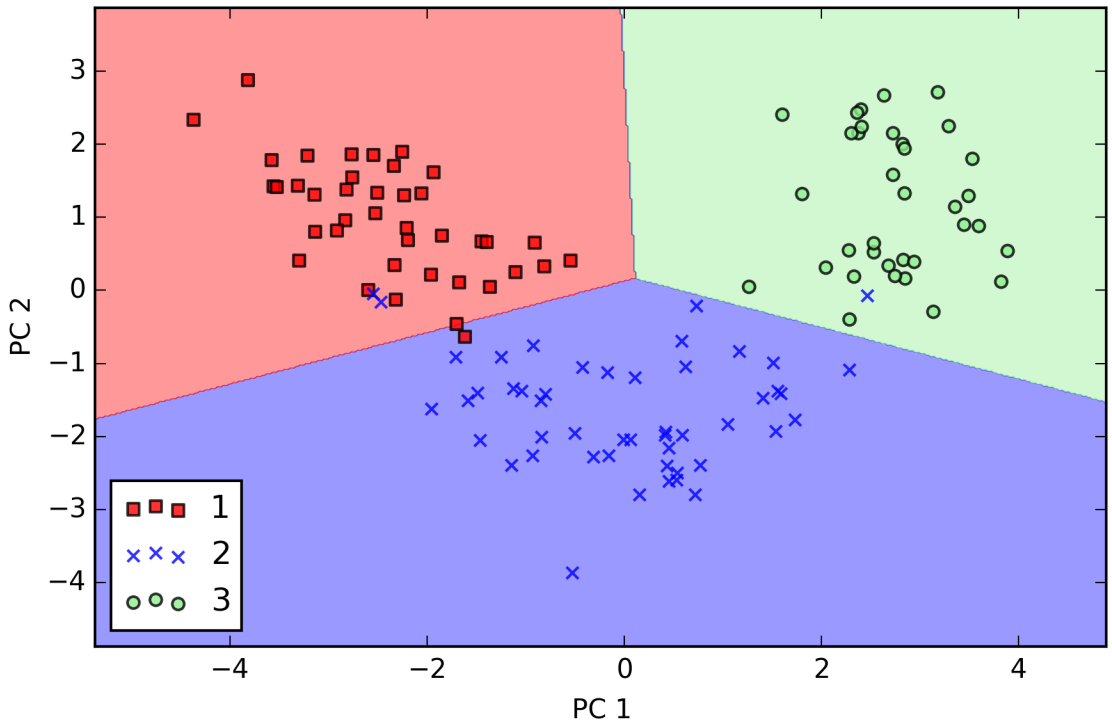
PCA .: passos

1. Estandardizar o *dataset*;
2. Construir a matriz de covariância;
3. Decompor a matriz de covariância em seus autovetores e autovalores;
4. Selecionar k autovetores que correspondem aos k maiores autovalores;
5. Construir matriz de projeção W a partir dos k autovetores;
6. Transformar o espaço de características de d para k usando W.

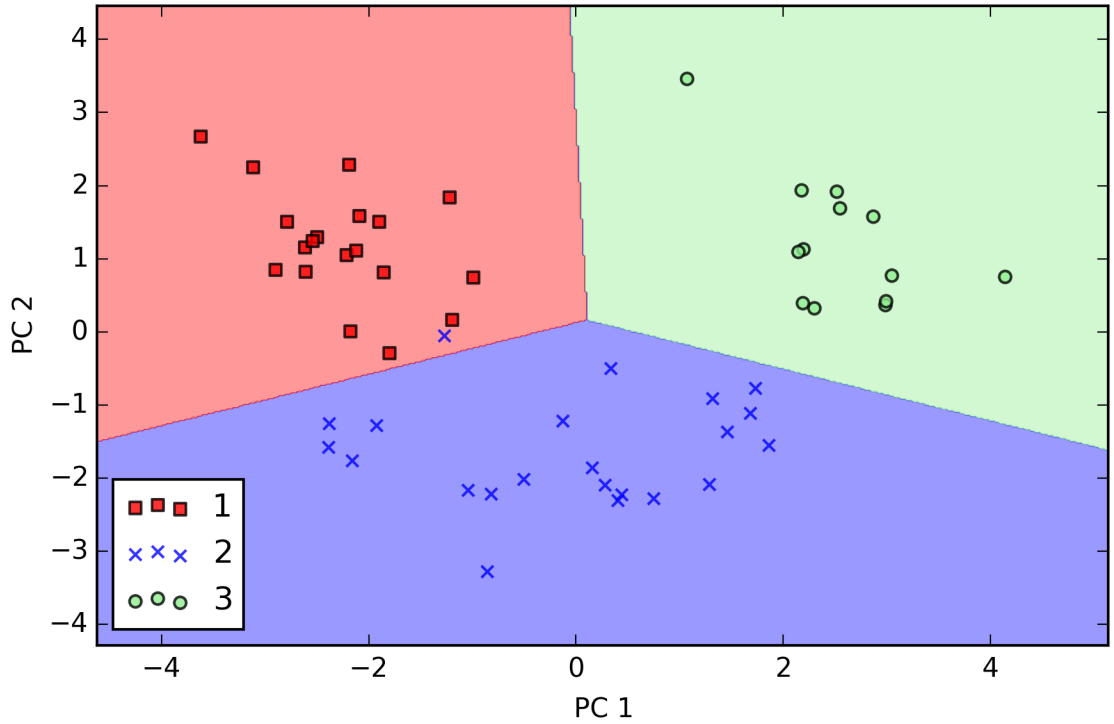
```
from sklearn.linear_model import LogisticRegression

lr = LogisticRegression()
lr = lr.fit(X_train_pca, y_train)
```

```
plot_decision_regions(X_train_pca, y_train, classifier=lr)
plt.xlabel('PC 1')
plt.ylabel('PC 2')
plt.legend(loc='lower left')
plt.tight_layout()
# plt.savefig('./figures/pca3.p
plt.show()
```



```
plot_decision_regions(X_test_pca, y_test, classifier=lr)
plt.xlabel('PC 1')
plt.ylabel('PC 2')
plt.legend(loc='lower left')
plt.tight_layout()
# plt.savefig('./figures/p
plt.show()
```



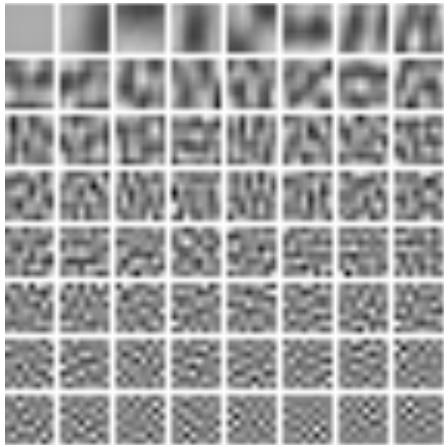
Exemplo para compressão de imagem

256 × 256 pixels

32 x 32 blocos de 8x8
(1024 vetores em $\Re^{64}$)

256 × 256 pixels

64 Componentes principais ordenadas



1ª , 2ª , 3ª , …
9ª , 10ª ,

…

4 componentes

8 componentes

16 componentes

32 componentes

Divulgação científica

http://www.saense.com.br
https://www.facebook.com/saense/

Hendrik Macedo
Escreve sobre Inteligência Artificial no Saense.
http://www.saense.com.br/autores/artigos-publicados-por-hendrik-macedo/