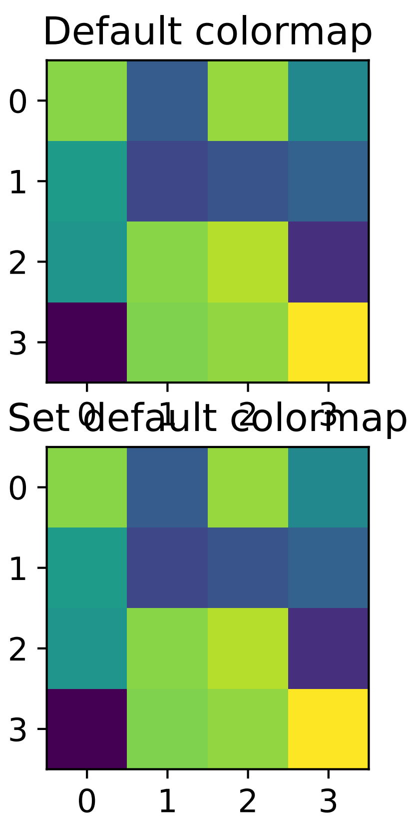
Lecture 03: PCA applications class demo

Contents

```
import os
import random
import sys

import numpy as np
import pandas as pd

sys.path.append(os.path.join(os.path.abspath(".."), "code"))

import matplotlib.pyplot as plt
import seaborn as sns
from plotting_functions import *
from sklearn.decomposition import PCA
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler

plt.rcParams["font.size"] = 10
plt.rcParams["figure.figsize"] = (5, 4)
%matplotlib inline
pd.set_option("display.max_colwidth", 0)

%config InlineBackend.figure_formats = ['svg']

DATA_DIR = os.path.join(os.path.abspath(".."), "data/")
```



PCA applications

There are tons of applications of PCA. In this section we'll look at a few example applications.

Big five personality traits

Have you heard of the **Big Five personality** traits:

- **Extroversion**: Reflects sociability, energy, and enthusiasm
- **Openness**: Measures creativity, curiosity, and willingness to try new experiences.
- **Agreeableness**: Represents compassion, kindness, and cooperativeness.
- **Conscientiousness**: Reflects discipline, organization, and reliability.
- **Neuroticism**: Measures emotional stability and tendency toward negative emotions.

These were identified using PCA based on a large-scale personality data set obtained from the Open-Source Psychometrics Project: https://openpsychometrics.org/.

This dataset contains several thousand responses to an online personality survey consisting of 50 statements rated on a 5-point likert scale.

You can see the statements themselves at this link.

```
## Big 5 data
bf = pd.read_csv("https://remiller1450.github.io/data/big5data.csv", sep='\t')

## Split the personality questions from the demographics
bf_demo = bf[['race','age','engnat','gender','hand','source','country']]
bf_demo
```

	race	age	engnat	gender	hand	source	country
0	3	53	1	1	1	1	US
1	13	46	1	2	1	1	US
2	1	14	2	2	1	1	PK
3	3	19	2	2	1	1	RO
4	11	25	2	2	1	2	US
...	...	...	...	...	...	...	...
19714	11	15	1	2	1	2	SG
19715	3	37	1	2	1	2	US
19716	5	16	2	1	1	2	US
19717	12	16	1	1	1	5	NG
19718	3	35	1	1	1	1	US

19719 rows × 7 columns

The rest of the columns are the responses to an online personality survey consisting of 50 statements rated on a 5-point likert scale, where

- 0=missed
- 1=Disagree
- 3=Neutral
- 5=Agree

Let’s examine these columns:

```
bf_qs = bf.drop(columns=['race','age','engnat','gender','hand','source','count
bf_qs
```

	E1	E2	E3	E4	E5	E6	E7	E8	E9	E10	...	O1	O2	O3	O4	O5
0	4	2	5	2	5	1	4	3	5	1	...	4	1	3	1	5
1	2	2	3	3	3	3	1	5	1	5	...	3	3	3	3	2
2	5	1	1	4	5	1	1	5	5	1	...	4	5	5	1	5
3	2	5	2	4	3	4	3	4	4	5	...	4	3	5	2	4
4	3	1	3	3	3	1	3	1	3	5	...	3	1	1	1	3
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
19714	1	4	3	5	4	3	1	2	1	5	...	1	3	5	3	4
19715	2	3	2	3	2	3	2	4	4	4	...	1	2	3	2	3
19716	2	5	4	5	5	5	1	2	1	5	...	5	3	1	3	4
19717	1	4	2	3	2	4	1	3	4	5	...	3	2	5	3	4
19718	2	3	1	5	3	3	3	2	2	4	...	5	1	5	1	4

19719 rows × 50 columns

Let's store the mapping of column names and actual questions from this link in a dictionary.

```
# Dictionary mapping feature names to actual questions
questions_dict = {
    "E1": "I am the life of the party.",
    "E2": "I don't talk a lot.",
    "E3": "I feel comfortable around people.",
    "E4": "I keep in the background.",
    "E5": "I start conversations.",
    "E6": "I have little to say.",
    "E7": "I talk to a lot of different people at parties.",
    "E8": "I don't like to draw attention to myself.",
    "E9": "I don't mind being the center of attention.",
    "E10": "I am quiet around strangers.",
    "N1": "I get stressed out easily.",
    "N2": "I am relaxed most of the time.",
    "N3": "I worry about things.",
    "N4": "I seldom feel blue.",
    "N5": "I am easily disturbed.",
    "N6": "I get upset easily.",
    "N7": "I change my mood a lot.",
    "N8": "I have frequent mood swings.",
    "N9": "I get irritated easily.",
    "N10": "I often feel blue.",
    "A1": "I feel little concern for others.",
    "A2": "I am interested in people.",
    "A3": "I insult people.",
    "A4": "I sympathize with others' feelings.",
    "A5": "I am not interested in other people's problems.",
    "A6": "I have a soft heart.",
    "A7": "I am not really interested in others.",
    "A8": "I take time out for others.",
    "A9": "I feel others' emotions.",
    "A10": "I make people feel at ease.",
    "C1": "I am always prepared.",
    "C2": "I leave my belongings around.",
    "C3": "I pay attention to details.",
    "C4": "I make a mess of things.",
    "C5": "I get chores done right away.",
    "C6": "I often forget to put things back in their proper place.",
    "C7": "I like order.",
    "C8": "I shirk my duties.",
    "C9": "I follow a schedule.",
    "C10": "I am exacting in my work.",
    "O1": "I have a rich vocabulary.",
    "O2": "I have difficulty understanding abstract ideas.",
    "O3": "I have a vivid imagination.",
    "O4": "I am not interested in abstract ideas.",
    "O5": "I have excellent ideas.",
    "O6": "I do not have a good imagination.",
    "O7": "I am quick to understand things.",
    "O8": "I use difficult words.",
    "O9": "I spend time reflecting on things.",
    "O10": "I am full of ideas."
}
```

```
bf_qs.shape
```

```
(19719, 50)
```

❓❓ Questions for you

- Why would applying PCA be useful in this scenario?

```
from sklearn.decomposition import PCA
pca_bfq = PCA(n_components=5, random_state=42)
pca_bfq.fit(bf_qs)
```

```
▾ PCA
PCA(n_components=5, random_state=42)
```

```
pca_bfq.explained_variance_ratio_.sum()
```

```
np.float64(0.4645454150547112)
```

The first 5 components are coverting 46% of the information. Good to know!

```
# Get the components
W = pca_bfq.components_

# Get the feature names
feature_names = bf_qs.columns
```

```
# Create a DataFrame for better visualization
components_df = pd.DataFrame(W.T, columns=[f'PC{i}' for i in range(W.shape[0])

# Display the most influential features for each component
for i in range(W.shape[0]):
    print(f"\nTop positive features for PC{i}:\n")
    top_pos_features = components_df.iloc[:, i].sort_values(ascending=False).h
    for feature, value in top_pos_features.items():
        print(f"{feature}: {questions_dict.get(feature, 'Unknown question')} (

    print(f"\nTop negative features for PC{i}:\n")
    top_neg_features = components_df.iloc[:, i].sort_values(ascending=True).he
    for feature, value in top_neg_features.items():
        print(f"{feature}: {questions_dict.get(feature, 'Unknown question')} (

    print("\n" + "-"*50 + "\n")
```

```
Top positive features for PC0:

E7: I talk to a lot of different people at parties. (Score: 0.2635)
E3: I feel comfortable around people. (Score: 0.2526)
E5: I start conversations. (Score: 0.2409)
E9: I don't mind being the center of attention. (Score: 0.1923)
E1: I am the life of the party. (Score: 0.1869)
A10: I make people feel at ease. (Score: 0.1469)
A2: I am interested in people. (Score: 0.1409)
N2: I am relaxed most of the time. (Score: 0.1288)
C5: I get chores done right away. (Score: 0.1080)
N4: I seldom feel blue. (Score: 0.1079)

Top negative features for PC0:

E10: I am quiet around strangers. (Score: -0.2235)
N10: I often feel blue. (Score: -0.2222)
E4: I keep in the background. (Score: -0.2116)
N8: I have frequent mood swings. (Score: -0.2052)
N9: I get irritated easily. (Score: -0.1973)
E6: I have little to say. (Score: -0.1967)
E2: I don't talk a lot. (Score: -0.1956)
N6: I get upset easily. (Score: -0.1944)
N7: I change my mood a lot. (Score: -0.1854)
N1: I get stressed out easily. (Score: -0.1793)

--------------------------------------------------



Top positive features for PC1:

N8: I have frequent mood swings. (Score: 0.2663)
N7: I change my mood a lot. (Score: 0.2525)
N6: I get upset easily. (Score: 0.2462)
E7: I talk to a lot of different people at parties. (Score: 0.2176)
C6: I often forget to put things back in their proper place. (Score: 0.2085)
N1: I get stressed out easily. (Score: 0.2071)
N9: I get irritated easily. (Score: 0.2044)
C4: I make a mess of things. (Score: 0.2034)
E9: I don't mind being the center of attention. (Score: 0.1984)
C2: I leave my belongings around. (Score: 0.1973)

Top negative features for PC1:

E2: I don't talk a lot. (Score: -0.2271)
E8: I don't like to draw attention to myself. (Score: -0.1674)
E6: I have little to say. (Score: -0.1587)
E4: I keep in the background. (Score: -0.1483)
E10: I am quiet around strangers. (Score: -0.1358)
C5: I get chores done right away. (Score: -0.1231)
A7: I am not really interested in others. (Score: -0.1221)
N2: I am relaxed most of the time. (Score: -0.1111)
C1: I am always prepared. (Score: -0.1111)
A5: I am not interested in other people's problems. (Score: -0.1084)
```

```
--------------------------------------------------


Top positive features for PC2:

C9: I follow a schedule. (Score: 0.2718)
C5: I get chores done right away. (Score: 0.2397)
A6: I have a soft heart. (Score: 0.2313)
A4: I sympathize with others' feelings. (Score: 0.2306)
A9: I feel others' emotions. (Score: 0.2273)
C7: I like order. (Score: 0.2237)
N3: I worry about things. (Score: 0.2056)
N1: I get stressed out easily. (Score: 0.1985)
A8: I take time out for others. (Score: 0.1746)
C1: I am always prepared. (Score: 0.1689)

Top negative features for PC2:

C6: I often forget to put things back in their proper place. (Score: -0.2440)
C2: I leave my belongings around. (Score: -0.2286)
A5: I am not interested in other people's problems. (Score: -0.1914)
A3: I insult people. (Score: -0.1884)
A1: I feel little concern for others. (Score: -0.1869)
C8: I shirk my duties. (Score: -0.1696)
C4: I make a mess of things. (Score: -0.1506)
A7: I am not really interested in others. (Score: -0.1479)
N2: I am relaxed most of the time. (Score: -0.1454)
E9: I don't mind being the center of attention. (Score: -0.1270)

--------------------------------------------------


Top positive features for PC3:

O8: I use difficult words. (Score: 0.3518)
O1: I have a rich vocabulary. (Score: 0.3224)
O10: I am full of ideas. (Score: 0.2381)
O3: I have a vivid imagination. (Score: 0.2346)
O9: I spend time reflecting on things. (Score: 0.1939)
O5: I have excellent ideas. (Score: 0.1810)
O7: I am quick to understand things. (Score: 0.1777)
C2: I leave my belongings around. (Score: 0.1683)
C6: I often forget to put things back in their proper place. (Score: 0.1262)
E4: I keep in the background. (Score: 0.1185)

Top negative features for PC3:

O2: I have difficulty understanding abstract ideas. (Score: -0.3167)
O4: I am not interested in abstract ideas. (Score: -0.2953)
O6: I do not have a good imagination. (Score: -0.2328)
A1: I feel little concern for others. (Score: -0.1877)
C5: I get chores done right away. (Score: -0.1422)
E7: I talk to a lot of different people at parties. (Score: -0.1281)
E1: I am the life of the party. (Score: -0.1164)
E3: I feel comfortable around people. (Score: -0.1018)
N5: I am easily disturbed. (Score: -0.1016)
```

```
A5: I am not interested in other people's problems. (Score: -0.0977)

--------------------------------------------------


Top positive features for PC4:

O8: I use difficult words. (Score: 0.2359)
A5: I am not interested in other people's problems. (Score: 0.2298)
A1: I feel little concern for others. (Score: 0.2212)
A3: I insult people. (Score: 0.2068)
A7: I am not really interested in others. (Score: 0.1963)
N9: I get irritated easily. (Score: 0.1923)
C1: I am always prepared. (Score: 0.1904)
O1: I have a rich vocabulary. (Score: 0.1874)
C7: I like order. (Score: 0.1863)
C9: I follow a schedule. (Score: 0.1850)

Top negative features for PC4:

A4: I sympathize with others' feelings. (Score: -0.2255)
C6: I often forget to put things back in their proper place. (Score: -0.2194)
A6: I have a soft heart. (Score: -0.2133)
C2: I leave my belongings around. (Score: -0.1963)
A9: I feel others' emotions. (Score: -0.1712)
A8: I take time out for others. (Score: -0.1564)
A2: I am interested in people. (Score: -0.1366)
C4: I make a mess of things. (Score: -0.1145)
O2: I have difficulty understanding abstract ideas. (Score: -0.1105)
E8: I don't like to draw attention to myself. (Score: -0.1008)

--------------------------------------------------
```

Mapping PCA Components to Big Five Traits

PC	Label	Mapped Big Five Trait	Justification
PC0			
PC1			
PC2			
PC3			
PC4			

PCA for visualization

- One of the most common applications of PCA is visualizing high dimensional data.
- Suppose we want to visualize 20-dimensional [countries of the world data].
- The dataset has country names linked to population, area size, GDP, literacy percentage, birthrate, mortality, net migration etc.

```
df = pd.read_csv(DATA_DIR + "countries of the world.csv")
df.head()
```

	Country	Region	Population	Area (sq. mi.)	Pop. Density (per sq. mi.)	Coastline (coast/area ratio)	Net migration
0	Afghanistan	ASIA (EX. NEAR EAST)	31056997	647500	48,0	0,00	23,06
1	Albania	EASTERN EUROPE	3581655	28748	124,6	1,26	-4,93
2	Algeria	NORTHERN AFRICA	32930091	2381740	13,8	0,04	-0,39
3	American Samoa	OCEANIA	57794	199	290,4	58,29	-20,71
4	Andorra	WESTERN EUROPE	71201	468	152,1	0,00	6,6

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 227 entries, 0 to 226
Data columns (total 20 columns):
 #   Column                              Non-Null Count  Dtype  
---  ------                              --------------  -----  
 0   Country                             227 non-null    object 
 1   Region                              227 non-null    object 
 2   Population                          227 non-null    int64  
 3   Area (sq. mi.)                      227 non-null    int64  
 4   Pop. Density (per sq. mi.)          227 non-null    object 
 5   Coastline (coast/area ratio)        227 non-null    object 
 6   Net migration                       224 non-null    object 
 7   Infant mortality (per 1000 births)  224 non-null    object 
 8   GDP ($ per capita)                  226 non-null    float64
 9   Literacy (%)                        209 non-null    object 
 10  Phones (per 1000)                   223 non-null    object 
 11  Arable (%)                          225 non-null    object 
 12  Crops (%)                           225 non-null    object 
 13  Other (%)                           225 non-null    object 
 14  Climate                             205 non-null    object 
 15  Birthrate                           224 non-null    object 
 16  Deathrate                           223 non-null    object 
 17  Agriculture                         212 non-null    object 
 18  Industry                            211 non-null    object 
 19  Service                             212 non-null    object 
dtypes: float64(1), int64(2), object(17)
memory usage: 35.6+ KB
```

```
X_countries = df.drop(columns=["Country", "Region"])
```

Let’s replace commas with periods in columns with type `object`.

```
def convert_values(value):
    value = str(value)
    value = value.replace(",", ".")
    return float(value)


for col in X_countries.columns:
    if X_countries[col].dtype == object:
        X_countries[col] = X_countries[col].apply(convert_values)
```

```
X_countries.head()
```

	Population	Area (sq. mi.)	Pop. Density (per sq. mi.)	Coastline (coast/area ratio)	Net migration	Infant mortality (per 1000 births)	GDP ($ per capita)	Lite
0	31056997	647500	48.0	0.00	23.06	163.07	700.0	
1	3581655	28748	124.6	1.26	-4.93	21.52	4500.0	
2	32930091	2381740	13.8	0.04	-0.39	31.00	6000.0	
3	57794	199	290.4	58.29	-20.71	9.27	8000.0	
4	71201	468	152.1	0.00	6.60	4.05	19000.0	

- We have missing values
- The features are in different scales.
- Let's create a pipeline with `SimpleImputer` and `StandardScaler`.

```
from sklearn.impute import SimpleImputer
n_components = 2
pipe = make_pipeline(SimpleImputer(), StandardScaler(), PCA(n_components=n_com
pipe.fit(X_countries)
X_countries_pca = pipe.transform(X_countries)
```

```
print(
    "Variance Explained by the first %d principal components: %0.3f percent"
    % (n_components, sum(pipe.named_steps["pca"].explained_variance_ratio_) *
)
```

```
Variance Explained by the first 2 principal components: 43.583 percent
```

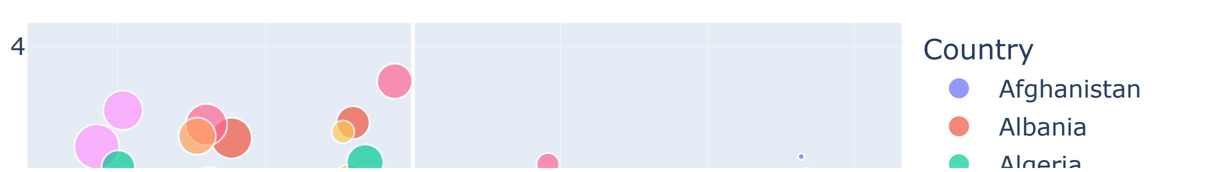
- Good to know!

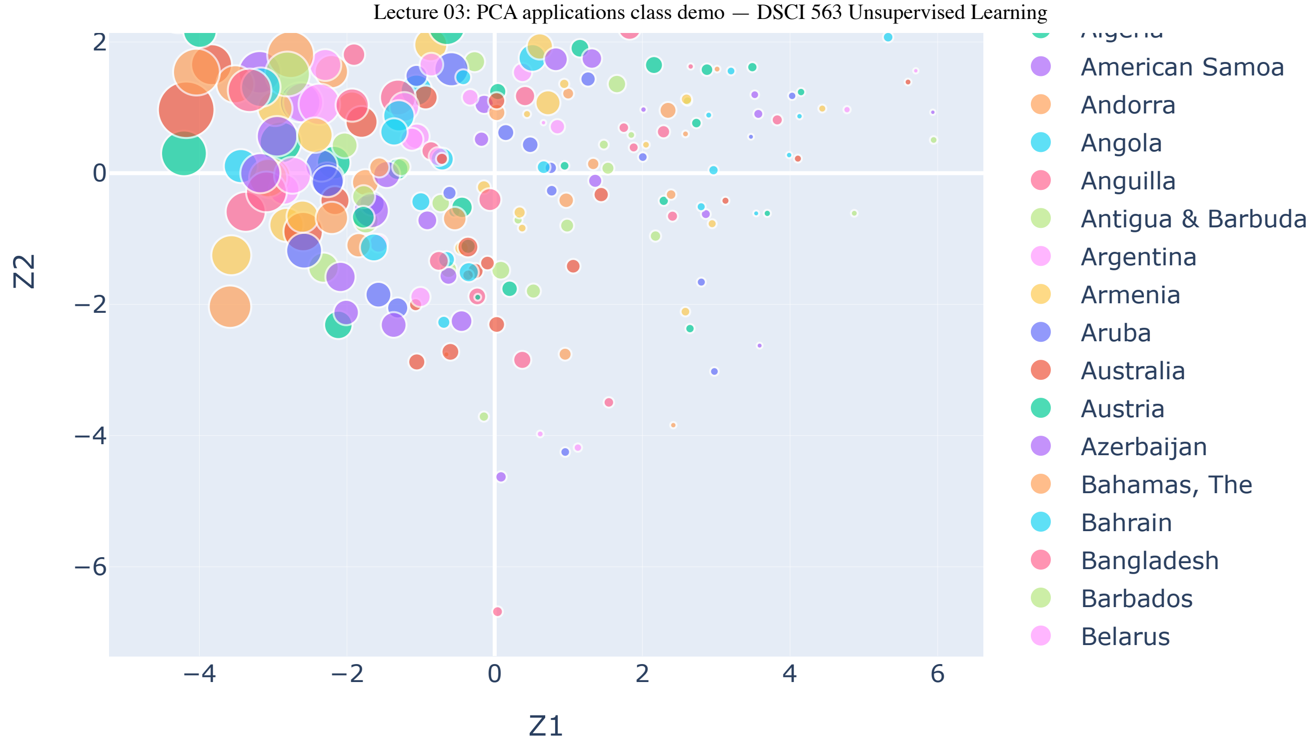
For each example, let's get other information from the original data.

```
pca_df = pd.DataFrame(X_countries_pca, columns=["Z1", "Z2"], index=X_countries
pca_df["Country"] = df["Country"]
pca_df["Population"] = X_countries["Population"]
pca_df["GDP"] = X_countries["GDP ($ per capita)"]
pca_df["Crops"] = X_countries["Crops (%)"]
pca_df["Infant mortality"] = X_countries["Infant mortality (per 1000 births)"]
pca_df["Birthrate"] = X_countries["Birthrate"]
pca_df["Literacy"] = X_countries["Literacy (%)"]
pca_df["Net migration"] = X_countries["Net migration"]
pca_df.fillna(pca_df["GDP"].mean(), inplace=True)
pca_df.head()
```

	Z1	Z2	Country	Population	GDP	Crops	Infant mortality	Bir
0	5.259255	2.326683	Afghanistan	31056997	700.0	0.22	163.07	
1	-0.260777	-1.491964	Albania	3581655	4500.0	4.42	21.52	
2	1.154648	1.904628	Algeria	32930091	6000.0	0.25	31.00	
3	-0.448853	-2.255437	American Samoa	57794	8000.0	15.00	9.27	
4	-2.211518	1.547689	Andorra	71201	19000.0	0.00	4.05	

```
import plotly.express as px

fig = px.scatter(
    pca_df,
    x="Z1",
    y="Z2",
    color="Country",
    size="GDP",
    hover_data=[
        "Population",
        "Infant mortality",
        "Literacy",
        "Birthrate",
        "Net migration",
    ],
)
fig.show()
```

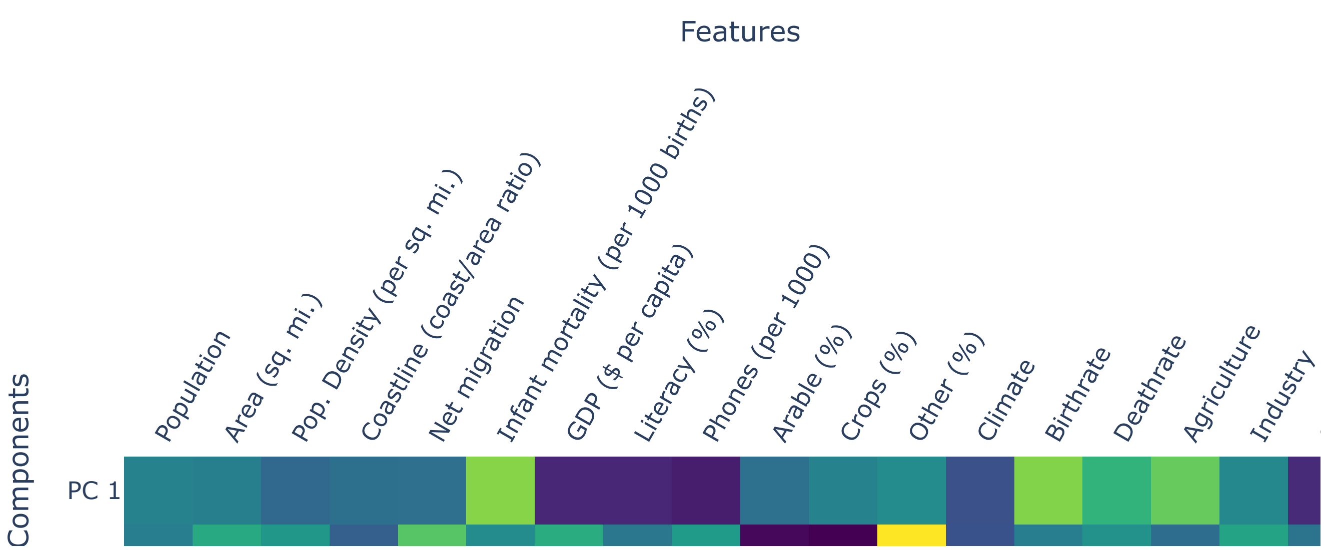




How to interpret the components?

- Each principal component has a coefficient associated with each feature in our original dataset.
- We can interpret the components by looking at the features with relatively bigger values (in magnitude) for coefficients for each components.

```
component_labels = ["PC " + str(i + 1) for i in range(n_components)]
W = pipe.named_steps["pca"].components_
plot_pca_w_vectors(W, component_labels, X_countries.columns)
```



(Optional) PCA for compression

One way to think of PCA is that it's a **data compression** algorithm. Let's work through compressing a random image from the internet using PCA.

> **Note**
>
> For this demo, we will be working with grayscale images. You can use PCA for coloured images as well. Just that it is a bit more work, as you have to apply it separately for each colour channel.

```
from matplotlib.pyplot import imread, imshow

# source: https://www.amazon.ca/Reflection-Needlework-Cross-Stitch-Embroidery-
img = imread(os.path.join('../img/cats_reflection.jpg'))
plt.figure(figsize=[6,4])
plt.axis('off')
image_bw = img.sum(axis=2)/img.sum(axis=2).max()
print('dimensions:', image_bw.shape)
plt.imshow(image_bw, cmap=plt.cm.gray)
plt.show()
```

```
dimensions: (879, 580)
```

Let’s apply PCA with 40 components.

```
n_components=40
pca=PCA(n_components=n_components)
pca.fit(image_bw)
```

```
▾ PCA
PCA(n_components=40)
```

We can examine the components.

```
pca.components_.shape
```

```
(40, 580)
```

We can also call SVD on our own and check whether the components of sklearn match with what’s returned by SVD.

```
image_bw_centered = image_bw - np.mean(image_bw, axis=0) # Let's center the im
U, S, Vt = np.linalg.svd(image_bw_centered, full_matrices=False)
U.shape, S.shape, Vt.shape
```

```
((879, 580), (580,), (580, 580))
```

Do the components given by `sklearn` match with the rows of `Vt`?

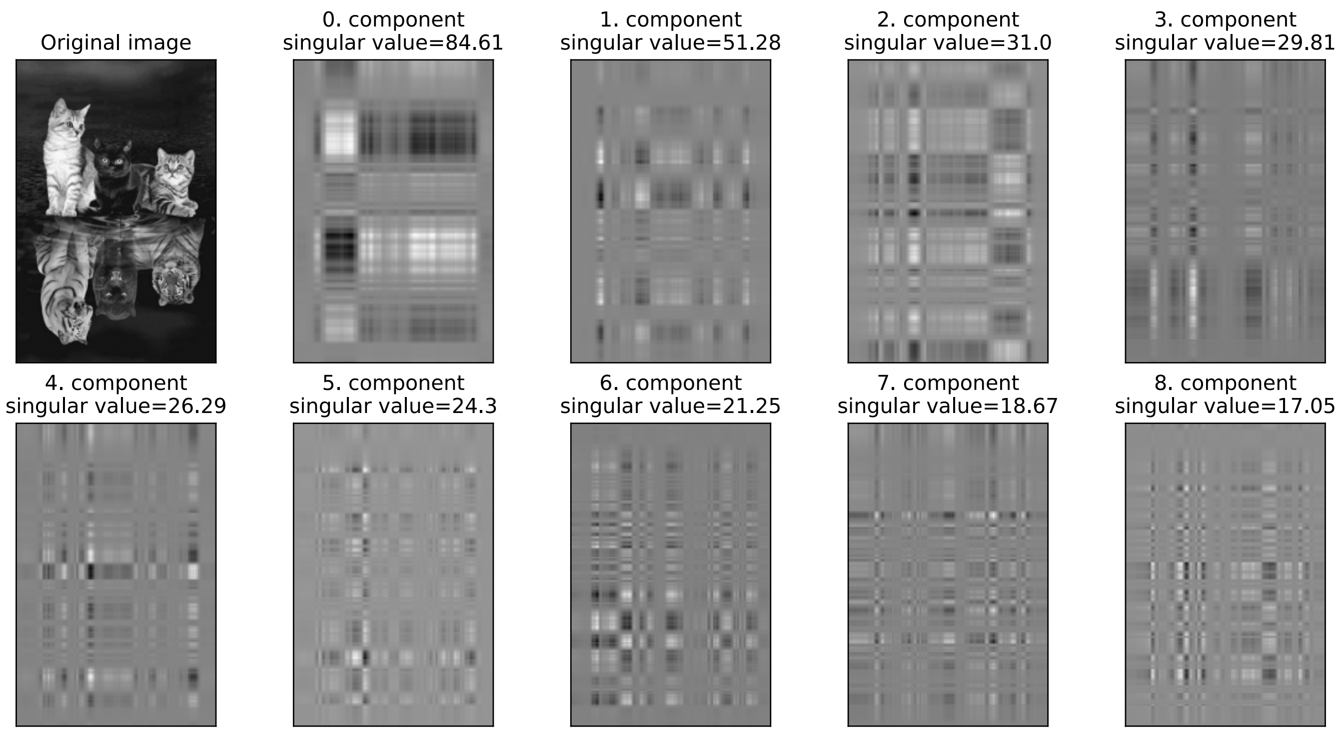
```
np.allclose(abs(pca.components_[0]), abs(Vt[0])) # taking abs because the solu
```

```
True
```

Let's explore the component images created by the first few components:
$S_0 U_0 Vt_0 + S_1 U_1 Vt_1 + \dots + S_8 U_8 Vt_8 + \dots$

```
components = []
k = 10
for i in range(k):
    components.append(U[:, i][:, None]@Vt[i, :][None,:])
```

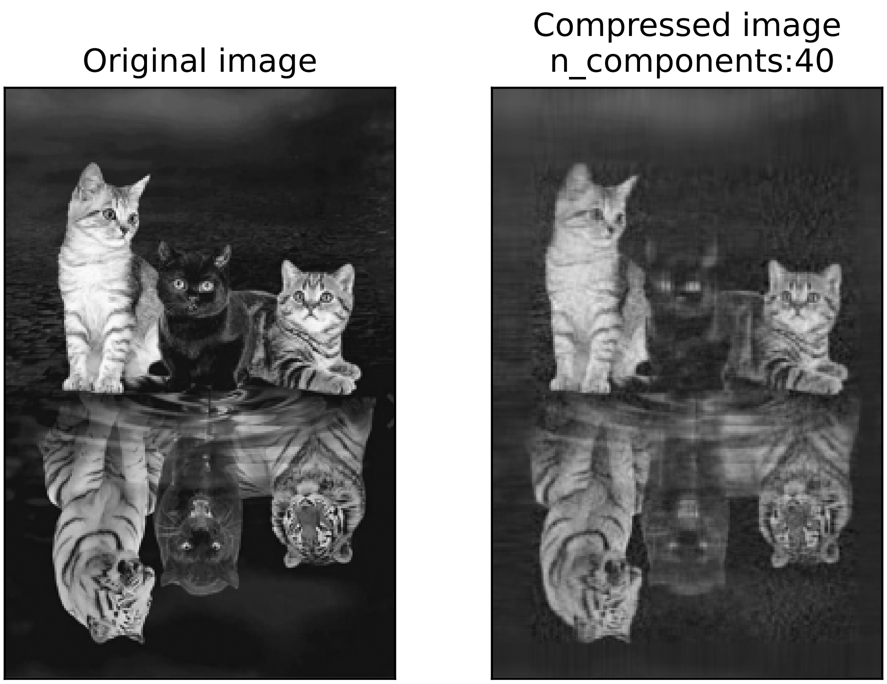
```
fig, axes = plt.subplots(2, 5, figsize=(14, 7), subplot_kw={"xticks": (), "yti
for i, (component, ax) in enumerate(zip(components, axes.ravel())):
    if i==0:
        ax.imshow(image_bw, cmap=plt.cm.gray)
        ax.set_title('Original image')
    else:
        ax.imshow(component, cmap=plt.cm.gray)
        ax.set_title(f"{i-1}. component\nsingular value={np.round(S[i-1],2)}")
plt.show()
```



The first component with the largest singular value seems to be capturing the overall brightness and contrast and the subsequent components seem to be capturing more details such as textures and edges.

How good is the reconstruction with just 40 components?

```
Z_image = pca.transform(image_bw)
W_image = pca.components_
X_hat_image = pca.inverse_transform(Z_image)
plot_orig_compressed(image_bw, X_hat_image, n_components)
```



Pretty good reconstruction considering that we have only 40 components out of original 580 components.

Why is this compression?

- The size of the original matrix X: ___
- The size of the matrices decomposed matrices U, S, and V^T after applying SVD: ___
- The size of the matrices U, S, and V^T after compression: ___

```
n, d = image_bw.shape[0], image_bw.shape[1]
n, d
```

```
(879, 580)
```

```
U.shape
```

```
(879, 580)
```

```
S.shape
```

```
(580,)
```

```
Vt.shape
```

```
(580, 580)
```

Let's truncate for dimensionality reduction.

```
Z_svd = ((U[:, :n_components] * S[:n_components]))
```

```
Z_svd.shape
```

```
(879, 40)
```

```
W_svd = Vt[:n_components,:]
```

```
W_svd.shape
```

```
(40, 580)
```

```
orig_size = (n*d) # How many numbers do you need to store for the original ima
orig_size
```

```
509820
```

```
# How many numbers do you need to store for the compressed image?
# n * n_components to store U
# n_components to store S
# n_components*d to store Vt
compressed_size = (n*n_components) + n_components + (n_components*d)
compressed_size
```

```
58400
```

Dimensionality reduction to reduce overfitting in supervised setting

- Often you would see dimensionality reduction being used as a preprocessing step in supervised learning setup.
- More features means higher possibility of overfitting.
- If we reduce number of dimensions, it may reduce overfitting and computational complexity.

Dimensionality reduction for anomaly detection

- A common application for dimensionality reduction is anomaly or outliers detection. For example:
  - Detecting fraud transactions.
  - Detecting irregular activity in video frames.
  - It's hard to find good anomaly detection datasets. A popular one is [The KDD Cup '99 dataset](#).


Burglary