Load Data

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

url = "https://raw.githubusercontent.com/rfordatascience/tidytues
age_gaps = pd.read_csv(url)

age_gaps.columns = age_gaps.columns.str.lower()

# parse dates
date_cols = [c for c in age_gaps.columns if "birthdate" in c]
for c in date_cols:
    age_gaps[c] = pd.to_datetime(age_gaps[c], errors="coerce")

age_gaps.info()
age_gaps.head()
```

Show hidden output

Next steps: Generate code with age_gaps | New interactive sheet

Data Processing

```
age_gaps.isna().sum()
```

	0
movie_name	0
release_year	0
director	0
age_difference	0
couple_number	0
actor_1_name	0
actor_2_name	0
character_1_gender	0
character_2_gender	0
actor_1_birthdate	0
actor_2_birthdate	0
actor_1_age	0
actor_2_age	0

dtype: int64

```
1 age_gaps.describe()
```

	release_year	age_difference	couple_number	actor_1_birthdate	a
count	1155.000000	1155.000000	1155.000000	1155	
mean	2000.799134	10.424242	1.398268	1960-09-07 07:15:07.012987008	
min	1935.000000	0.000000	1.000000	1889-04-16 00:00:00	
25%	1997.000000	4.000000	1.000000	1953-05-16 00:00:00	
50%	2004.000000	8.000000	1.000000	1964-10-03 00:00:00	
75%	2012.000000	15.000000	2.000000	1973-08-07 12:00:00	
max	2022.000000	52.000000	7.000000	1996-06-01 00:00:00	
std	16.365819	8.511086	0.754419	NaN	

I add a new feature:

- release_to_2026
- younger_age
- order_age
- older_gender

- meam_age
- relative_age

```
# copy age_gaps
age_gaps_copy = age_gaps.copy()
# drop column = 'movie_name', 'director', 'couple_numbe
age_gaps_copy = age_gaps_copy.drop(columns=['movie_name
# add a new column named "release to 2026" (2026 - rele
age_gaps_copy['release_to_2026'] = 2026 - age_gaps_copy

# compare actor_1_age, actor_2_age from each row and as
age_gaps_copy['younger_age'] = age_gaps_copy[['actor_1_
age_gaps_copy['order_age'] = age_gaps_copy[['actor_1_ag

age_gaps_copy['older_gender'] = np.where(
    age_gaps_copy['actor_1_age'] > age_gaps_copy['actor
    np.where(
        age_gaps_copy['actor_2_age'] > age_gaps_copy['a
        'same_age'))

# add a column named 'mean_age' = (younger_age+order_ag
age_gaps_copy['mean_age'] = (age_gaps_copy['younger_age

# add new column named " relative_age_gap"
age_gaps_copy['relative_age_gap'] = age_gaps_copy['age_

age_gaps_clean = age_gaps_copy.drop(columns=['actor_1_a
age_gaps_clean.head().reset_index()
```

	index	release_year	age_difference	couple_number	character_1_gend
0	0	1971	52	1	wom
1	1	2006	50	1	m
2	2	2002	49	1	m
3	3	1998	45	1	m
4	4	2010	43	1	m

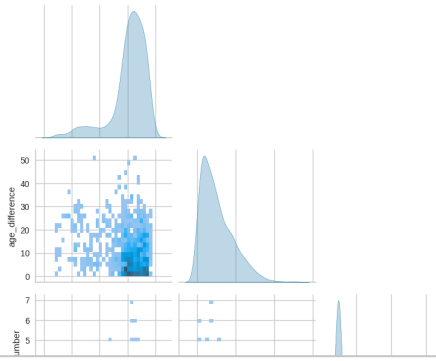
Next steps: Generate code with age_gaps_clean | New interactive sheet

EDA

```
import seaborn as sns
import matplotlib.pyplot as plt
```

```
sns.pairplot(age_gaps_clean, kind="hist", diag_kind="kde", corner
```

```
<seaborn.axisgrid.PairGrid at 0x7ecbde3dfd70>
```



Question

Is there a threshold in age gap size beyond which gender ordering becomes systematically asymmetric, indicating a shift in how relationships are portrayed?

Feature Normalization

Because the ranges of the features differ substantially and the units are not consistent, feature normalization was applied to prevent any single feature from dominating the distance calculations

```
from sklearn.preprocessing import StandardScaler


feature_cols = ['mean_age', 'relative_age_gap']

X = age_gaps_copy[feature_cols]

sc = StandardScaler()
X_scaled = sc.fit_transform(X)

X_scaled_df = pd.DataFrame(
    X_scaled,
    columns=feature_cols,
    index=age_gaps_copy.index
)

X_scaled_df.head()
```

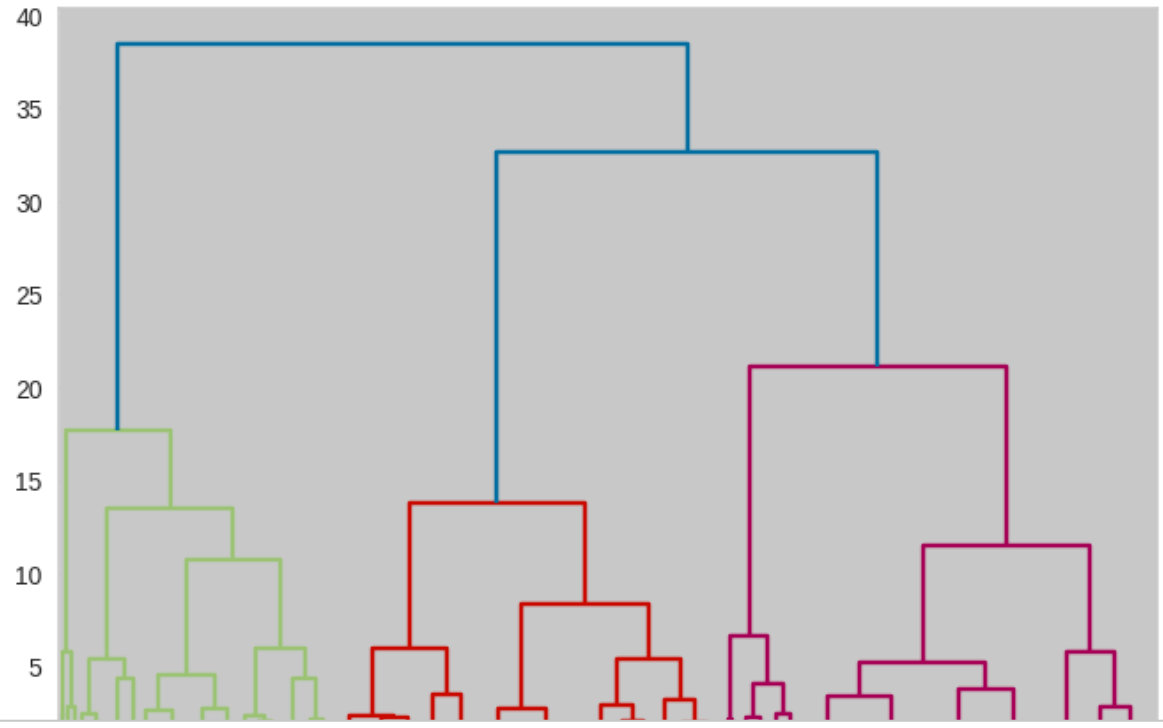
	mean_age	relative_age_gap
0	1.693628	5.471972
1	1.693628	4.956601
2	1.132271	6.020999
3	1.257017	4.588479
4	3.003460	2.193749

Next steps: Generate code with X_scaled_df | New interactive sheet

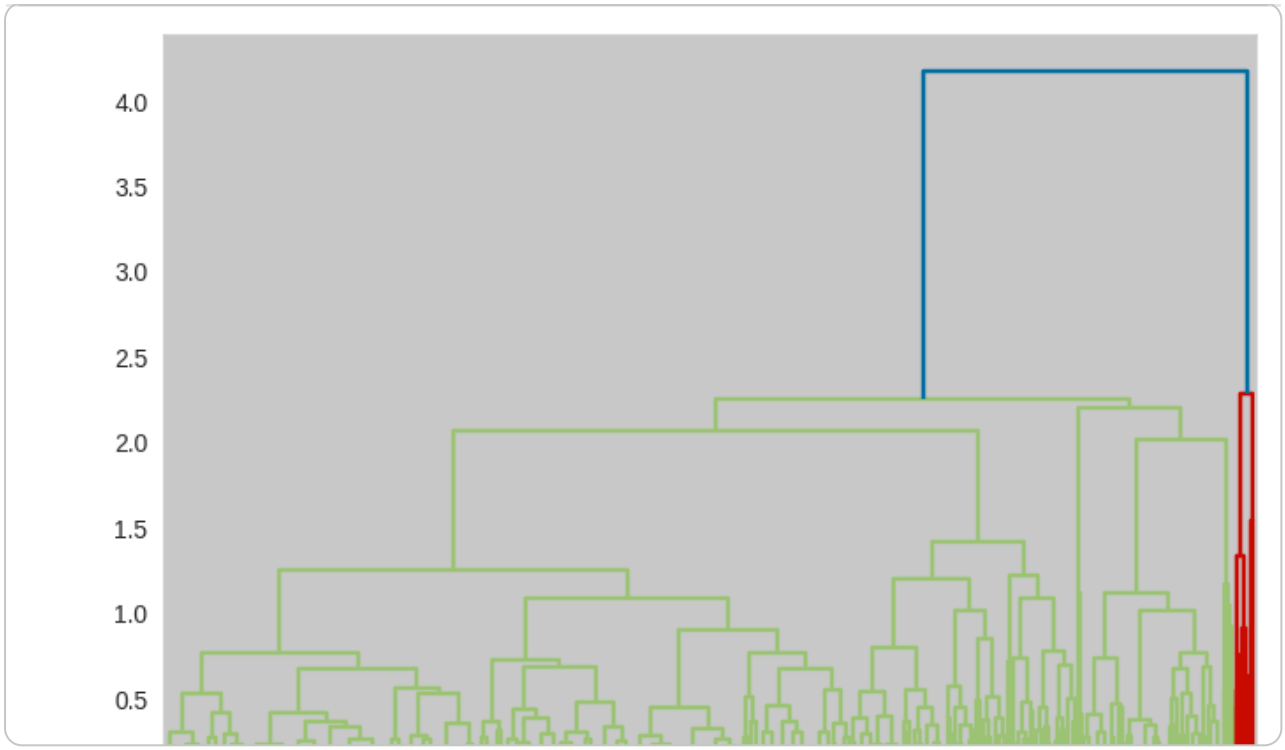
Hierarchical Clustering

```
data_hiera = age_gaps_copy.copy()
```

```
from scipy.cluster.hierarchy import dendrogram
from scipy.cluster.hierarchy import linkage, fcluster

Z_ward = linkage(X_scaled, 'ward') #You may try different methods
tree_2 = dendrogram(Z_ward,)
```



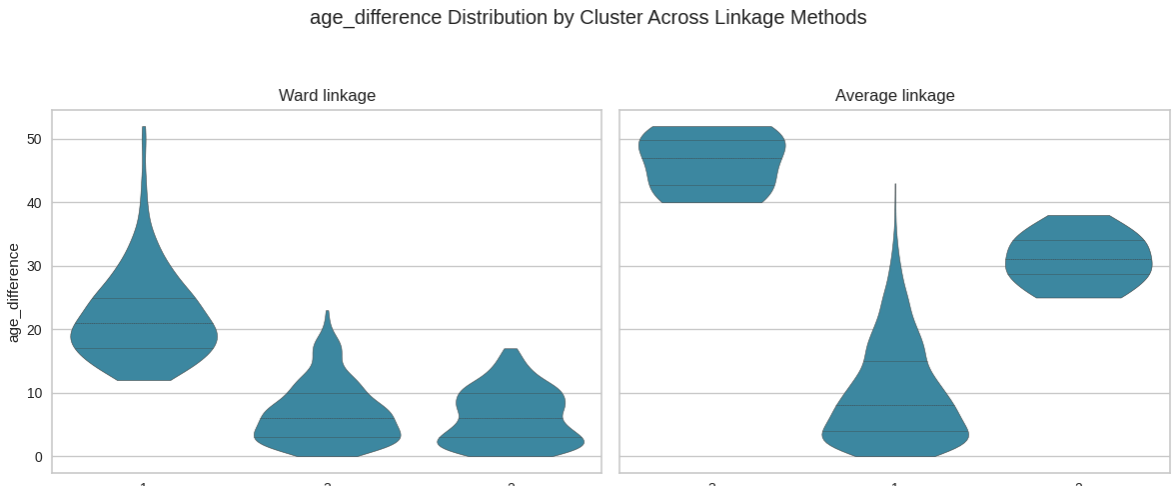

```
Z_average = linkage(X_scaled, 'average') #You may try different m
tree_2 = dendrogram(Z_average,)
```




```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from scipy.cluster.hierarchy import linkage, fcluster


df = age_gaps_copy.copy()
y_var = "age_difference"
feature_cols = ["mean_age", "relative_age_gap"]
n_clusters = 3
work_df = df.dropna(subset=feature_cols + [y_var]).copy()

scaler = StandardScaler()
X_scaled = scaler.fit_transform(work_df[feature_cols].to_numpy())

Z_ward = linkage(X_scaled, method="ward")
Z_avg  = linkage(X_scaled, method="average")

work_df["cluster_ward"] = fcluster(Z_ward, t=n_clusters, criterio
work_df["cluster_avg"]  = fcluster(Z_avg,  t=n_clusters, criterio

fig, axes = plt.subplots(1, 2, figsize=(12, 5), sharey=True)

linkages = [
    ("cluster_ward", "Ward linkage"),
    ("cluster_avg", "Average linkage"),
]

for ax, (col, title) in zip(axes, linkages):
    sns.violinplot(
        data=work_df,
        x=col,
        y=y_var,
```

```
        inner="quartile",
        cut=0,
        scale="width",
        color="#2c92b1",
        ax=ax
    )
    ax.set_title(title)
    ax.set_xlabel("Cluster")

axes[0].set_ylabel(y_var)

plt.suptitle(
    f"{y_var} Distribution by Cluster Across Linkage Methods",
    y=1.05
)
plt.tight_layout()
plt.show()
```

```
/tmp/ipython-input-3457659015.py:31: FutureWarning:

The `scale` parameter has been renamed and will be removed in v0.15.0.
  sns.violinplot(
/tmp/ipython-input-3457659015.py:31: FutureWarning:

The `scale` parameter has been renamed and will be removed in v0.15.0.
  sns.violinplot(
```



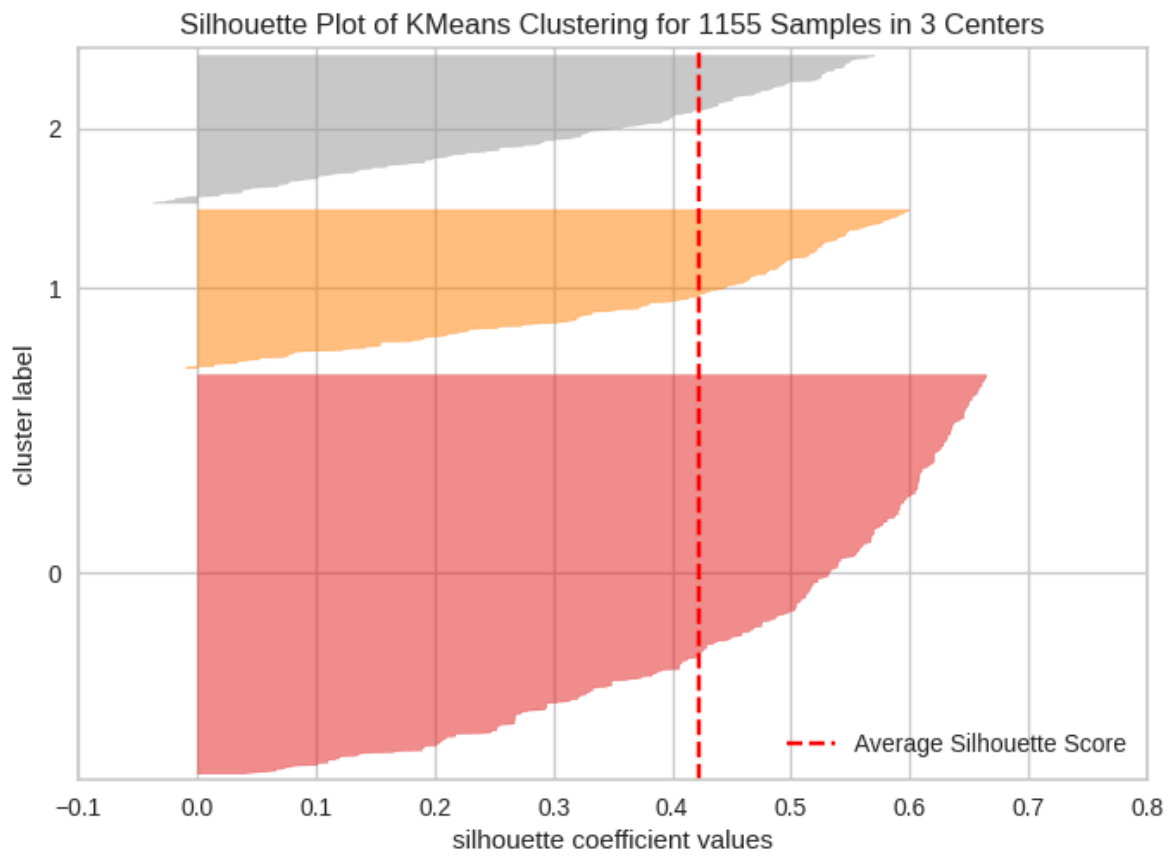
Silhuette Score

Try different number of clusters to see which number of clusters has higher score.

I tried number of 2, 3 and 4. 3 has best performance.

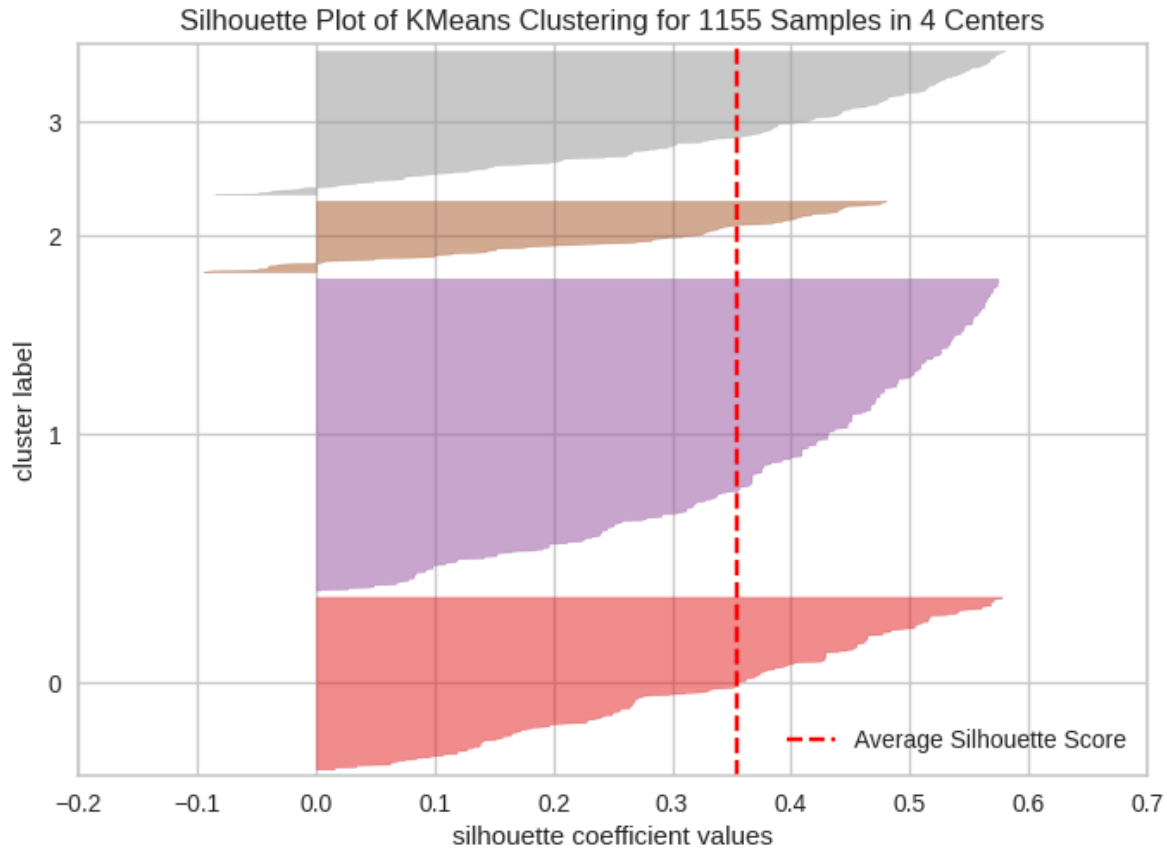
```
from sklearn.cluster import KMeans
from yellowbrick.cluster import KElbowVisualizer
from yellowbrick.cluster import SilhouetteVisualizer
n_clusters_kmeans = 3

kmeans_model = KMeans(n_clusters_kmeans, random_state=42)
visualizer = SilhouetteVisualizer(kmeans_model, timings=False)

```

```
visualizer.fit(X_scaled)        # Fit the data to the visualizer
visualizer.show()
```




```
<Axes: title={'center': 'Silhouette Plot of KMeans Clustering for 1155
```

```
from sklearn.cluster import KMeans
from yellowbrick.cluster import KElbowVisualizer
from yellowbrick.cluster import SilhouetteVisualizer
n_clusters_kmeans = 4

kmeans_model = KMeans(n_clusters_kmeans, random_state=42)
visualizer = SilhouetteVisualizer(kmeans_model, timings=False)

visualizer.fit(X_scaled)        # Fit the data to the visualizer
visualizer.show()
```




```
<Axes: title={'center': 'Silhouette Plot of KMeans Clustering for 1155
```

```
from sklearn.cluster import KMeans
from yellowbrick.cluster import KElbowVisualizer
from yellowbrick.cluster import SilhouetteVisualizer
n_clusters_kmeans = 2

kmeans_model = KMeans(n_clusters_kmeans, random_state=4
visualizer = SilhouetteVisualizer(kmeans_model, timings

visualizer.fit(X_scaled)        # Fit the data to the v
visualizer.show()
```


```
1 n_clusters = 3
2 data_hiera['cluster_labels'] = fcluster(Z_average, n_cl
3 data_hiera
```

	release_year	age_difference	couple_number	character_1_gender
0	1971	52	1	woman
1	2006	50	1	man
2	2002	49	1	man
3	1998	45	1	man
4	2010	43	1	man
...	...	...	...	...
1150	2013	0	1	man
1151	2006	0	1	woman
1152	2019	0	1	woman
1153	2007	0	1	woman
1154	2015	0	3	man

1155 rows × 14 columns


```
<Axes: title={'center': 'Silhouette Plot of KMeans Clustering for 1155
```

Next steps: Generate code with data_hiera | New interactive sheet

```
1 data_hiera['cluster_labels'].value_counts()
```

cluster_labels	count
1	1133
2	16
3	6

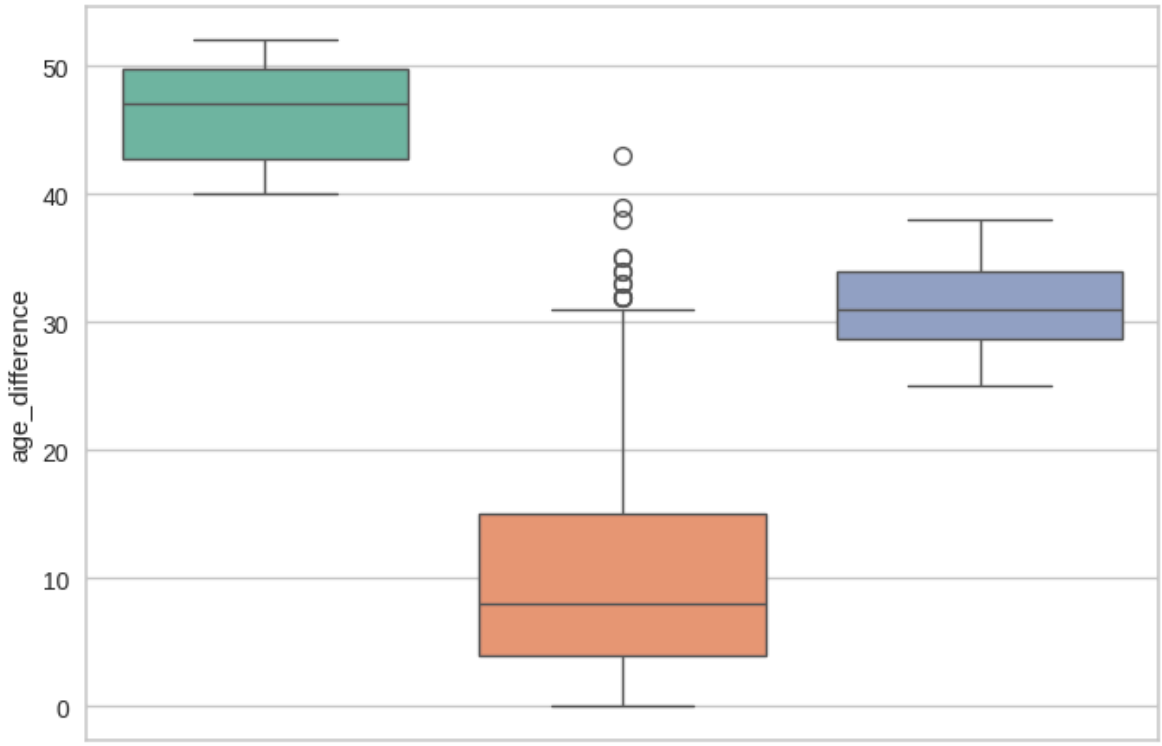
dtype: int64

```
1 data_hiera.groupby('cluster_labels')['age_difference'].describe()
```

cluster_labels	count	mean	std	min	25%	50%	75%	max
1	1133.0	9.943513	7.777061	0.0	4.00	8.0	15.00	43.0
2	16.0	31.000000	4.016632	25.0	28.75	31.0	34.00	38.0
3	6.0	46.333333	4.760952	40.0	42.75	47.0	49.75	52.0

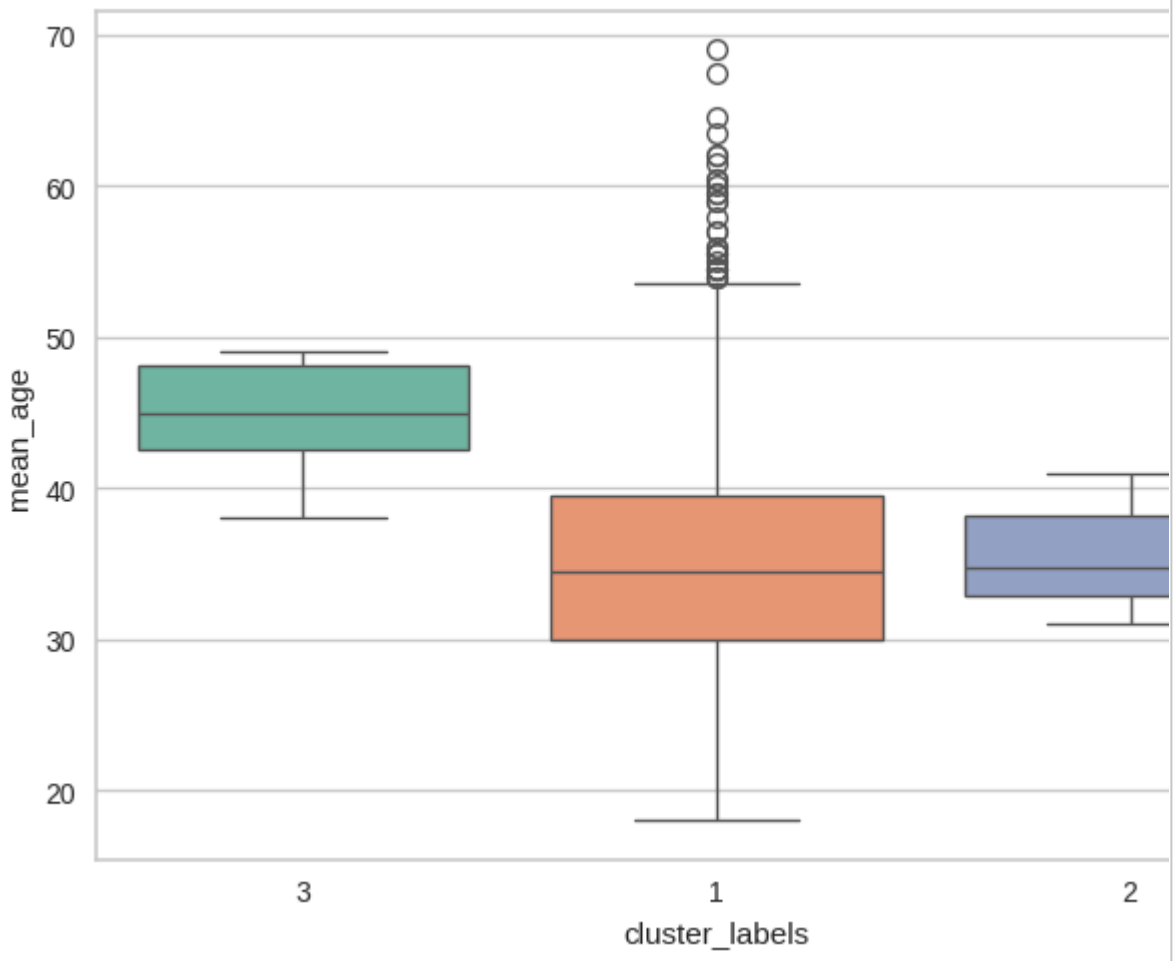
```
1 sns.boxplot(data=data_hiera, x='cluster_labels', y="age_differenc
```

```
/tmp/ipython-input-375687973.py:1: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be re

  sns.boxplot(data=data_hiera, x='cluster_labels', y="age_difference",
<Axes: xlabel='cluster_labels', ylabel='age_difference'>
```

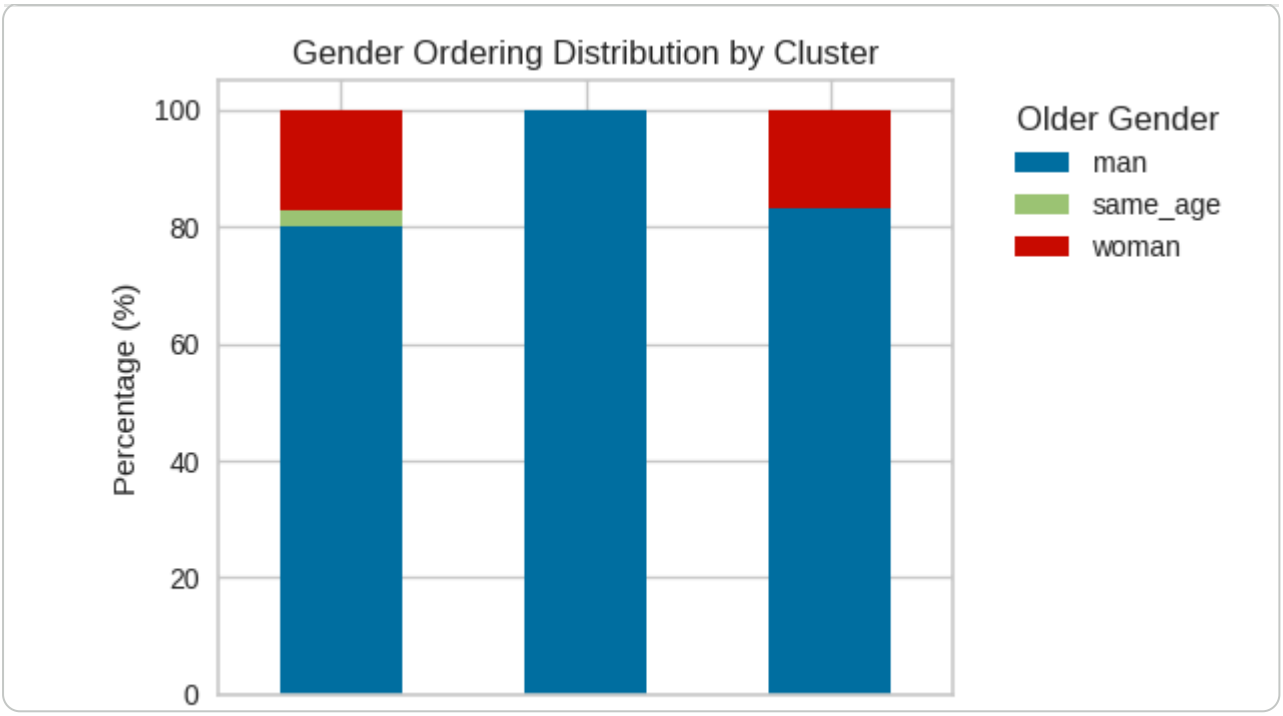



```
1 sns.boxplot(data=data_hiera, x='cluster_labels', y="mean_age", pa
```

```
/tmp/ipython-input-924421717.py:1: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be re

  sns.boxplot(data=data_hiera, x='cluster_labels', y="mean_age", palet
<Axes: xlabel='cluster_labels', ylabel='mean_age'>
```




```
import matplotlib.pyplot as plt

gender_pct = (
    pd.crosstab(data_hiera['cluster_labels'], data_hiera['older_g
    * 100
)

gender_pct.plot(
    kind='bar',
    stacked=True,
    figsize=(6, 4)
)

plt.ylabel('Percentage (%)')
plt.xlabel('Cluster')
plt.title('Gender Ordering Distribution by Cluster')
plt.legend(title='Older Gender', bbox_to_anchor=(1.05, 1), loc='u
plt.tight_layout()
plt.show()
```

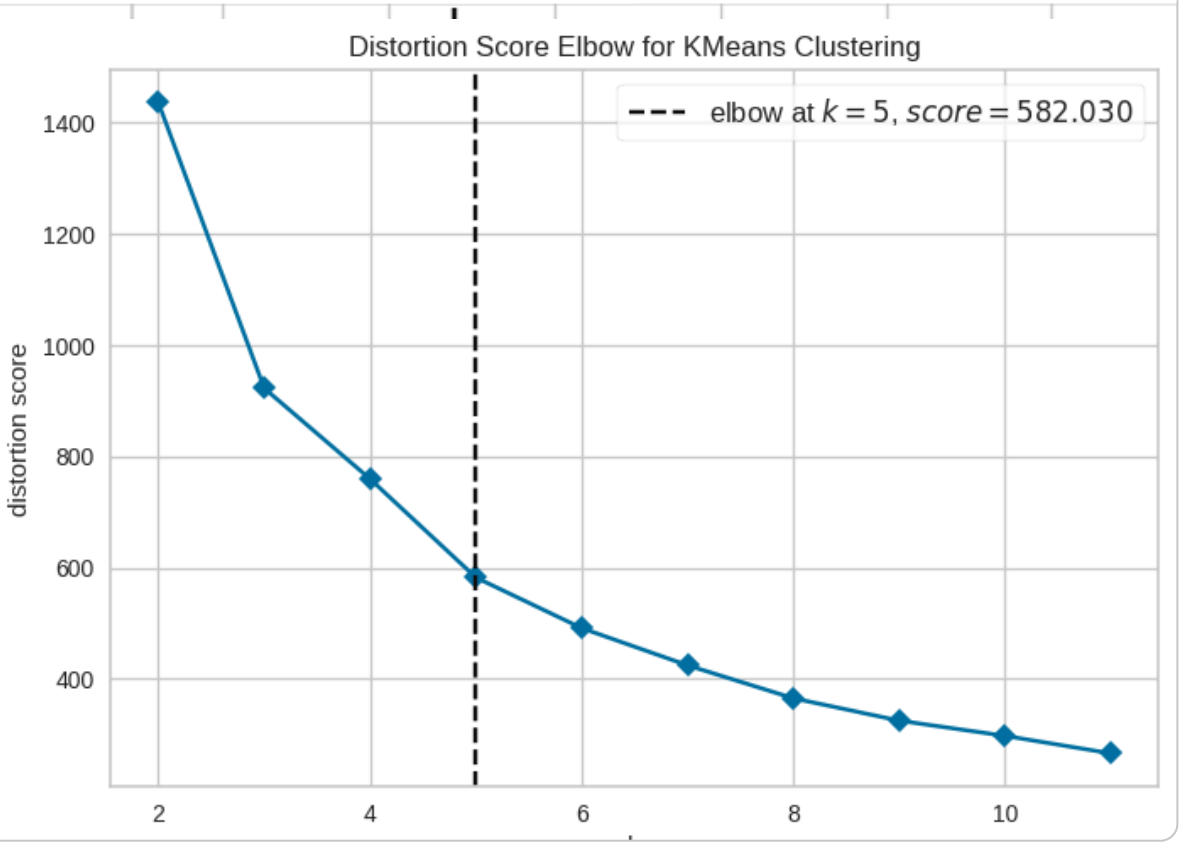


K-Mean

```
X_scaled_k=X_scaled_df.drop(columns=['cluster_labels'])
```

```
from sklearn.cluster import KMeans
from yellowbrick.cluster import KElbowVisualizer

clustering = KMeans(random_state=42)
visualizer = KElbowVisualizer(
    clustering, k=(2, 20), metric='distortion', timings
)

visualizer.fit(X_scaled_k)        # Fit the data to the
visualizer.show()
```


```
from sklearn.cluster import KMeans
from yellowbrick.cluster import KElbowVisualizer

clustering = KMeans(random_state=42)
visualizer = KElbowVisualizer(
    clustering, k=(2, 12), metric='distortion', timings=False, ra
)

visualizer.fit(X_scaled_k)        # Fit the data to the visualize
visualizer.show()
```




```
n_clusters_kmeans = 4
```

```
data_k=age_gaps_copy.copy()
data_k=data_k.drop(columns=['character_1_gender',   'character_2_
```

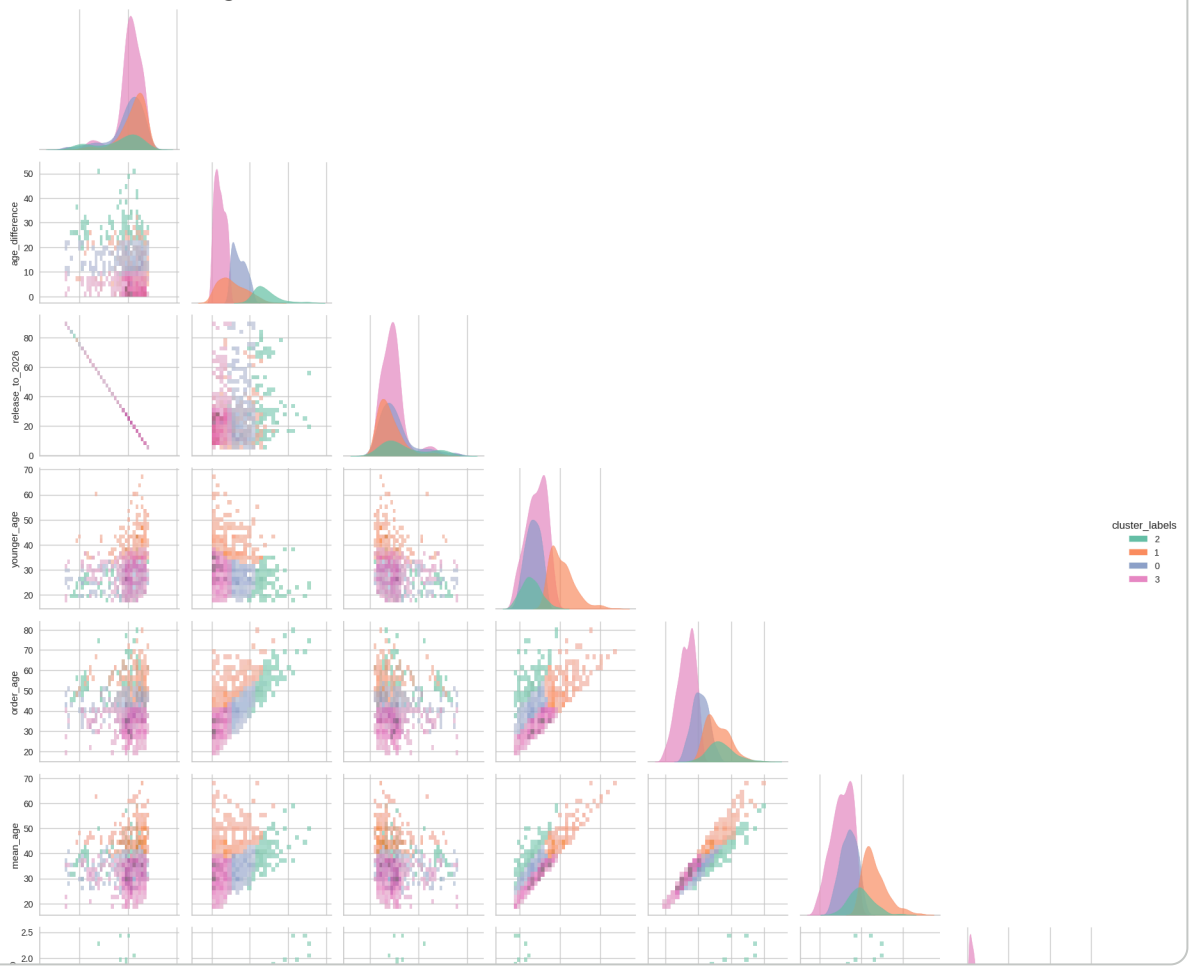
```
from sklearn.cluster import KMeans

# Initialize KMeans model with n_clusters and random_state=42
kmeans = KMeans(n_clusters=n_clusters_kmeans, random_state=42) # 

# Fit the model and predict cluster labels, excluding the existin
data_k['cluster_labels'] = kmeans.fit_predict(X_scaled_df).astype

# Display the value counts of the new cluster labels
data_k['cluster_labels'].value_counts().sort_index()
```

cluster_labels	count
0	279
1	238
2	112
3	526

dtype: int64

```
sns.pairplot(data_k, hue="cluster_labels", kind="hist", diag_kind=
    plot_kws={'alpha': 0.7}, # Adjusts transparency of the scatte
    diag_kws={'alpha': 0.7}  # Adjusts transparency of the diagon
    )
```

```
<seaborn.axisgrid.PairGrid at 0x7ecbdb578620>
```




```
1 data_k
```

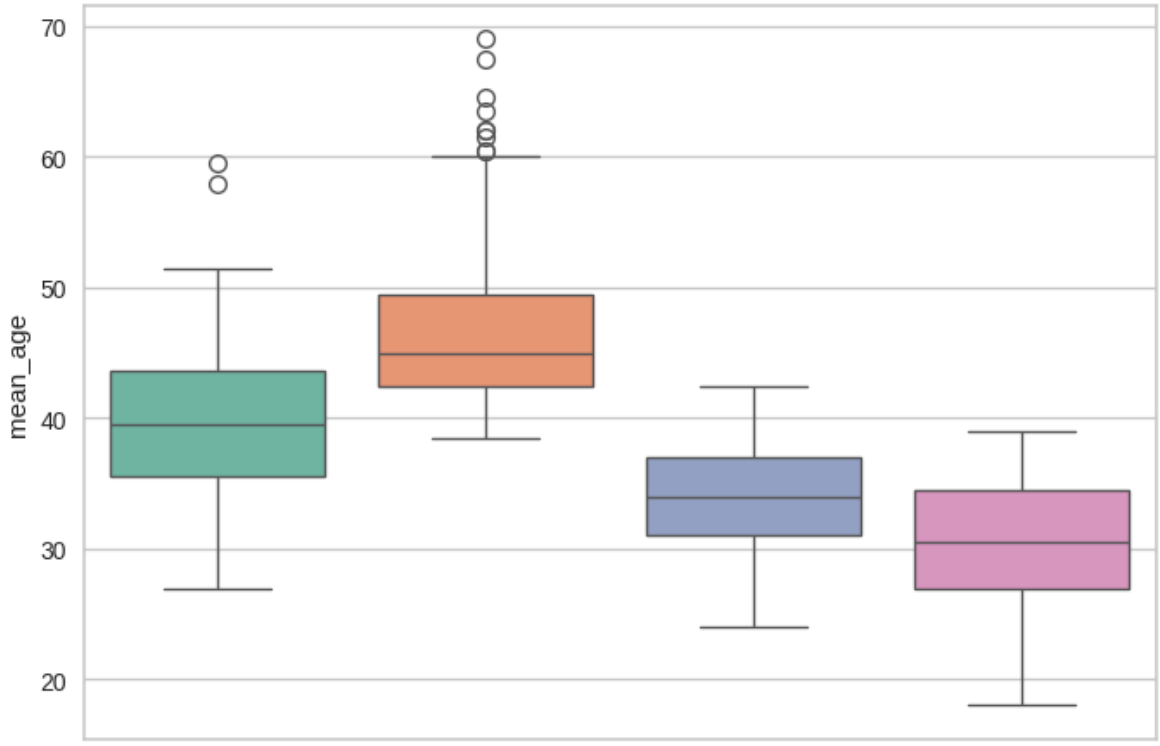
	release_year	age_difference	release_to_2026	younger_age	order_
0	1971	52	55	23	
1	2006	50	20	24	
2	2002	49	24	20	
3	1998	45	28	23	
4	2010	43	16	38	
...	...	...	...	...	
1150	2013	0	13	23	
1151	2006	0	20	42	
1152	2019	0	7	30	
1153	2007	0	19	21	
1154	2015	0	11	36	

1155 rows × 9 columns

Next steps: Generate code with data_k | New interactive sheet

```
1 sns.boxplot(data=data_k, x='cluster_labels', y="mean_age", palett
```

```
/tmp/ipython-input-1587560851.py:1: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be re

  sns.boxplot(data=data_k, x='cluster_labels', y="mean_age", palette="
<Axes: xlabel='cluster_labels', ylabel='mean_age'>
```




```
1 sns.boxplot(data=data_k, x='cluster_labels', y="relative_age_gap"
```

```
/tmp/ipython-input-988000534.py:1: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be re
```

```
  1 sns.boxplot(data=data_k, x='cluster_labels', y="order_age", palet
```

```
<Axes: xlabel='cluster_labels', ylabel='relative_age_gap'>
/tmp/ipython-input-21855133.py:1: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be re

  sns.boxplot(data=data_k, x='cluster_labels', y="order_age", palette=
<Axes: xlabel='cluster_labels', ylabel='order_age'>
```