a03-svd-results

November 14, 2024

1 Machine Learning - Assignment 3

Arne Huckemann (ahuckema), Elise Wolf (eliwolf)

```python
[41]: import numpy as np
      from numpy.linalg import svd, matrix_rank
      import pandas as pd
      import matplotlib.pyplot as plt
      from IPython import get_ipython
      from util import (
          svdcomp,
          nextplot,
          plot_matrix,
          plot_xy,
          plot_cov,
          match_categories,
      )  # see util.py
      from sklearn.cluster import KMeans

      # setup plotting
      import psutil
      inTerminal = not "IPKernelApp" in get_ipython().config
      inJupyterNb = any(filter(lambda x: x.endswith("jupyter-notebook"), psutil.
        ↪Process().parent().cmdline()))
      inJupyterLab = any(filter(lambda x: x.endswith("jupyter-lab"), psutil.Process().
        ↪parent().cmdline()))
      if not inJupyterLab:
          from IPython import get_ipython
          get_ipython().run_line_magic("matplotlib", "" if inTerminal else "notebook"␣
        ↪if inJupyterNb else "widget")
```

2 1 Intuition on SVD

```python
[42]: M1 = np.array(
          [
              [1, 1, 1, 0, 0],
              [1, 1, 1, 0, 0],
```

```python
        [1, 1, 1, 0, 0],
        [0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0],
    ]
)

M2 = np.array(
    [
        [0, 0, 0, 0, 0],
        [0, 2, 1, 2, 0],
        [0, 2, 1, 2, 0],
        [0, 2, 1, 2, 0],
        [0, 0, 0, 0, 0],
    ]
)

M3 = np.array([[0, 0, 0, 0], [0, 1, 1, 1], [0, 1, 1, 1], [0, 1, 1, 1], [0, 1,␣
 ↪1, 1]])

M4 = np.array(
    [
        [1, 1, 1, 0, 0],
        [1, 1, 1, 0, 0],
        [1, 1, 1, 0, 0],
        [0, 0, 0, 1, 1],
        [0, 0, 0, 1, 1],
    ]
)

M5 = np.array(
    [
        [1, 1, 1, 0, 0],
        [1, 1, 1, 0, 0],
        [1, 1, 1, 1, 1],
        [0, 0, 1, 1, 1],
        [0, 0, 1, 1, 1],
    ]
)

M6 = np.array(
    [
        [1, 1, 1, 1, 1],
        [1, 1, 1, 1, 1],
        [1, 1, 0, 1, 1],
        [1, 1, 1, 1, 1],
        [1, 1, 1, 1, 1],
    ]
```

```
)
```

2.1 1b

```
[43]: U1, s1, V1 = np.linalg.svd(M1)
      U2, s2, V2 = np.linalg.svd(M2)
      U3, s3, V3 = np.linalg.svd(M3)
      U4, s4, V4 = np.linalg.svd(M4)
      U5, s5, V5 = np.linalg.svd(M5)
      U6, s6, V6 = np.linalg.svd(M6)

      # Create diagonal matrices for each si
      S1 = np.diag(s1)
      S2 = np.diag(s2)
      S3 = np.diag(s3)
      S4 = np.diag(s4)
      S5 = np.diag(s5)
      S6 = np.diag(s6)

      # Print the results
      print("SVD of M1:")
      print("U1:", U1)
      print("S1:", S1)
      print("V1:", V1.T) # here we have to transpose again, since numpy gives V^T

      print("\nSVD of M2:")
      print("U2:", U2)
      print("S2:", S2)
      print("V2:", V2.T)

      print("\nSVD of M3:")
      print("U3:", U3)
      print("S3:", S3)
      print("V3:", V3.T)

      print("\nSVD of M4:")
      print("U4:", U4)
      print("S4:", S4)
      print("V4:", V4.T)

      print("\nSVD of M5:")
      print("U5:", U5)
      print("S5:", S5)
      print("V5:", V5.T)

      print("\nSVD of M6:")
      print("U6:", U6)
```

```
print("S6:", S6)
print("V6:", V6.T)
```

SVD of M1:
U1: [[-5.77350269e-01 8.16496581e-01 -8.75605293e-17 0.00000000e+00
    0.00000000e+00]
 [-5.77350269e-01 -4.08248290e-01 -7.07106781e-01 0.00000000e+00
    0.00000000e+00]
 [-5.77350269e-01 -4.08248290e-01 7.07106781e-01 0.00000000e+00
    0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
    1.00000000e+00]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 1.00000000e+00
    0.00000000e+00]]
S1: [[3.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
  0.00000000e+00]
 [0.00000000e+00 1.33801123e-16 0.00000000e+00 0.00000000e+00
  0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 1.31940794e-49 0.00000000e+00
  0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
  0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
  0.00000000e+00]]
V1: [[-0.57735027 0.81649658 -0.   0.   0.   ]
 [-0.57735027 -0.40824829 0.70710678 0.   0.   ]
 [-0.57735027 -0.40824829 -0.70710678 0.   0.   ]
 [-0.   0.   -0.   0.   1.   ]
 [-0.   0.   -0.   1.   0.   ]]

SVD of M2:
U2: [[0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
    1.00000000e+00]
 [-5.77350269e-01 8.16496581e-01 8.75605293e-17 0.00000000e+00
    0.00000000e+00]
 [-5.77350269e-01 -4.08248290e-01 -7.07106781e-01 0.00000000e+00
    0.00000000e+00]
 [-5.77350269e-01 -4.08248290e-01 7.07106781e-01 0.00000000e+00
    0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 1.00000000e+00
    0.00000000e+00]]
S2: [[5.19615242e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
  0.00000000e+00]
 [0.00000000e+00 2.44286311e-16 0.00000000e+00 0.00000000e+00
  0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 7.15081846e-48 0.00000000e+00
  0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
```

```
   0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
  0.00000000e+00]]
V2: [[-0.          0.         -0.          0.          1.        ]
 [-0.66666667  0.74535599 -0.          0.          0.        ]
 [-0.33333333 -0.2981424   0.89442719  0.          0.        ]
 [-0.66666667 -0.59628479 -0.4472136   0.          0.        ]
 [-0.          0.         -0.          1.          0.        ]]

SVD of M3:
U3: [[ 0.          0.          0.          1.          0.        ]
 [-0.5        -0.5        -0.5         0.         -0.5       ]
 [-0.5         0.83333333 -0.16666667  0.         -0.16666667]
 [-0.5        -0.16666667  0.83333333  0.         -0.16666667]
 [-0.5        -0.16666667 -0.16666667  0.          0.83333333]]
S3: [[3.46410162 0.         0.         0.        ]
 [0.         0.         0.         0.        ]
 [0.         0.         0.         0.        ]
 [0.         0.         0.         0.        ]]
V3: [[-0.          0.          0.          1.        ]
 [-0.57735027  0.81649658  0.          0.        ]
 [-0.57735027 -0.40824829 -0.70710678  0.        ]
 [-0.57735027 -0.40824829  0.70710678  0.        ]]

SVD of M4:
U4: [[-5.77350269e-01  0.00000000e+00  8.16496581e-01  0.00000000e+00
  -8.75605293e-17]
 [-5.77350269e-01  0.00000000e+00 -4.08248290e-01  0.00000000e+00
  -7.07106781e-01]
 [-5.77350269e-01  0.00000000e+00 -4.08248290e-01  0.00000000e+00
   7.07106781e-01]
 [ 0.00000000e+00 -7.07106781e-01  0.00000000e+00 -7.07106781e-01
   0.00000000e+00]
 [ 0.00000000e+00 -7.07106781e-01  0.00000000e+00  7.07106781e-01
   0.00000000e+00]]
S4: [[3.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
  0.00000000e+00]
 [0.00000000e+00 2.00000000e+00 0.00000000e+00 0.00000000e+00
  0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 1.33801123e-16 0.00000000e+00
  0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 1.43493693e-17
  0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
  1.31940794e-49]]
V4: [[-0.57735027 -0.          0.81649658  0.         -0.        ]
 [-0.57735027 -0.         -0.40824829  0.          0.70710678]
 [-0.57735027 -0.         -0.40824829  0.         -0.70710678]
```

```
 [-0.          -0.70710678  0.          -0.70710678 -0.          ]
 [-0.          -0.70710678  0.           0.70710678 -0.          ]]

SVD of M5:
U5: [[-3.94102719e-01 -5.00000000e-01  3.07706105e-01  7.07106781e-01
   -6.90381060e-17]
 [-3.94102719e-01 -5.00000000e-01  3.07706105e-01 -7.07106781e-01
    9.22184114e-17]
 [-6.15412209e-01 -4.37175683e-17 -7.88205438e-01 -1.51090987e-17
   -2.31803054e-17]
 [-3.94102719e-01  5.00000000e-01  3.07706105e-01 -1.81051444e-16
   -7.07106781e-01]
 [-3.94102719e-01  5.00000000e-01  3.07706105e-01  4.09931606e-17
    7.07106781e-01]]
S5: [[3.56155281e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
   0.00000000e+00]
 [0.00000000e+00 2.00000000e+00 0.00000000e+00 0.00000000e+00
   0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 5.61552813e-01 0.00000000e+00
   0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 1.46139661e-17
   0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
   4.64495594e-49]]
V5: [[-3.94102719e-01 -5.00000000e-01 -3.07706105e-01 -7.07106781e-01
   -0.00000000e+00]
 [-3.94102719e-01 -5.00000000e-01 -3.07706105e-01  7.07106781e-01
   -2.31872909e-17]
 [-6.15412209e-01 -1.63976402e-17  7.88205438e-01  1.20459692e-16
    6.60128332e-17]
 [-3.94102719e-01  5.00000000e-01 -3.07706105e-01 -1.12092159e-16
    7.07106781e-01]
 [-3.94102719e-01  5.00000000e-01 -3.07706105e-01 -1.12092159e-16
   -7.07106781e-01]]

SVD of M6:
U6: [[-4.61939766e-01 -1.91341716e-01  8.57093429e-01  1.24059883e-01
    4.14729722e-17]
 [-4.61939766e-01 -1.91341716e-01 -4.02662589e-01  7.66722140e-01
    7.69039490e-17]
 [-3.82683432e-01  9.23879533e-01 -1.55954879e-16 -2.97505904e-17
    9.89839760e-18]
 [-4.61939766e-01 -1.91341716e-01 -2.27215420e-01 -4.45391011e-01
   -7.07106781e-01]
 [-4.61939766e-01 -1.91341716e-01 -2.27215420e-01 -4.45391011e-01
    7.07106781e-01]]
S6: [[4.82842712e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
   0.00000000e+00]
```

```
[0.00000000e+00 8.28427125e-01 0.00000000e+00 0.00000000e+00
 0.00000000e+00]
[0.00000000e+00 0.00000000e+00 1.90057833e-16 0.00000000e+00
 0.00000000e+00]
[0.00000000e+00 0.00000000e+00 0.00000000e+00 5.82050373e-19
 0.00000000e+00]
[0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 2.10408590e-51]]
V6: [[-4.61939766e-01  1.91341716e-01  8.65999615e-01  6.68331181e-03
 -0.00000000e+00]
 [-4.61939766e-01  1.91341716e-01 -2.94967625e-01  8.14244497e-01
   4.23034501e-17]
 [-3.82683432e-01 -9.23879533e-01  8.53379562e-17 -3.02617043e-18
  -2.06215842e-18]
 [-4.61939766e-01  1.91341716e-01 -2.85515995e-01 -4.10463904e-01
   7.07106781e-01]
 [-4.61939766e-01  1.91341716e-01 -2.85515995e-01 -4.10463904e-01
  -7.07106781e-01]]
```

2.1.1 Analysis of SVD Results vs. Expected Properties

Matrix M1 (3x3 Block of Ones) - **Expected Structure**: Rank-1, with a single, large singular value (= 3) representing the entire matrix pattern, as it has a uniform structure in its non-zero block. - **Observed Results**: - **Singular Value Distribution**: The SVD results show that only the first singular value is non-zero (3.0), with all others close to zero. This aligns with expectations for a perfect rank-1 structure. - **Matrix Representation**: The SVD decomposes the matrix with a single significant component, indicating that the SVD efficiently represents the repeated pattern with one mode of variation. - **Interpretation**: The strong match between expected and calculated results highlights SVD's ability to capture repeated structures with minimal components. This matrix serves as an ideal case for rank-1 SVD, demonstrating that if a pattern is consistently replicated, the SVD provides an optimal representation with the minimum required rank.

Matrix M2 (Central 3×5 Pattern) - **Expected Structure**: Rank-1, with a single high-magnitude singular value (5.2) due to the repetition of the central pattern vertically. - **Observed Results**: - **Singular Value Distribution**: The single significant singular value (close to 5.2) matches well with the expected structure, confirming the matrix's rank-1 nature. - **Approximation Characteristics**: The dominant singular value indicates that the central pattern has a single, dominant structure that can be expressed in one dimension. - **Interpretation**: The agreement here shows that SVD identifies consistent patterns across matrix rows, even if they are limited to specific blocks within the matrix. It reinforces that SVD captures major structural elements and ignores spatial positioning, focusing instead on repeated patterns.

Matrix M3 (4×4 Repeated Row Pattern) - **Expected Structure**: Rank-1 due to the row-wise repetition, with a single dominant singular value (3.46) representing the repeated rows. - **Observed Results**: - **Singular Value Distribution**: The singular value distribution reflects a single significant value around 3.46, validating that the row repetitions dominate the structure. - **Row vs. Column Dominance**: The decomposition shows that the dominant pattern is row-oriented, as SVD treats the matrix as having a single mode of variation aligned with the rows. - **Interpretation**: This case demonstrates that SVD captures directionality in repeated row

structures, where rank-1 accurately models consistent row values without distinguishing individual positions within the rows.

Matrix M4 (Two Distinct Blocks) - **Expected Structure**: Rank-2 due to the two disjoint blocks, expecting two prominent singular values. - **Observed Results**: - **Singular Values**: The results show two notable singular values, representing each block, with   (larger block) and   (smaller block). The first component captures most of the information due to the size of the larger block. - **Rank-1 Approximation**: While rank-2 is required for full reconstruction, the rank-1 approximation will predominantly reflect the larger block, with diminished representation of the smaller block. - **Interpretation**: SVD efficiently separates two structurally independent blocks, where the larger block contributes most to the rank-1 approximation. This suggests that SVD prioritizes based on the spatial extent or energy of sub-patterns, making it suitable for approximations when one structure dominates the matrix.

Matrix M5 (Overlapping Patterns Creating Higher Rank) - **Expected Structure**: Rank-3, since overlapping patterns introduce additional dimensions. - **Observed Results**: - **Singular Value Distribution**: The SVD shows three significant singular values, reflecting the matrix's complexity due to overlapping structures. - **Pattern Approximation**: The rank-1 and rank-2 approximations provide smoothed versions of the patterns, averaging overlapping regions rather than capturing exact structure. - **Interpretation**: SVD handles overlapping structures by distributing representation across multiple singular values, which implies that SVD prioritizes major spatial variations over exact binary patterns in cases of complex overlap. This is useful in scenarios where a balanced approximation of multiple overlapping structures is desired.

Matrix M6 (Full Matrix with a Central Hole) - **Expected Structure**: Rank-2, as the single missing element introduces additional variation across the matrix. - **Observed Results**: - **Smooth Approximation**: The rank-1 approximation produces a nearly uniform pattern, smoothing over the central discontinuity. The central gap only begins to appear in higher-rank approximations. - **Singular Value Distribution**: The largest singular value is substantial, with smaller values representing the discontinuity's influence. - **Interpretation**: SVD's smoothing over the central gap highlights its bias towards representing larger continuous structures, even when a small discontinuity exists. This characteristic can be beneficial for noise reduction, though it may obscure small structural details in low-rank approximations.

This analysis highlights several key aspects of SVD in matrix approximation: 1. **Dominant Patterns**: SVD captures large-scale, repeated structures in rank-1 form, effectively ignoring positional specificity for consistent row/column values. 2. **Rank and Approximation Quality**: When multiple distinct or overlapping patterns exist, higher ranks are necessary to capture the complete structure. Rank-1 and rank-2 approximations will prioritize larger, more prominent patterns over finer details. 3. **Smoothness vs. Precision**: SVD tends to smooth out minor discontinuities, making it suitable for noise reduction but potentially less precise in capturing sharp, isolated differences (as seen in Matrix M6).

Overall, SVD's behavior aligns well with expectations for matrices with uniform or repeated structures, where rank-1 captures the essence of the data. For matrices with distinct or overlapping patterns, higher ranks provide a more accurate representation, though low-rank approximations can still yield meaningful insights about the dominant structure.

2.2 1c

```
[44]: # You can use the functions svdcomp and plot_matrix from util.py
X1_1 = svdcomp(M1, range(1))
X1_2 = svdcomp(M2, range(1))
X1_3 = svdcomp(M3, range(1))
X1_4 = svdcomp(M4, range(1))
X1_5 = svdcomp(M5, range(1))
X1_6 = svdcomp(M6, range(1))

print("Rank-1 Approximation of M1:")
print(X1_1)
plot_matrix(X1_1)
plt.title("Rank-1 Approximation of M1")

print("Rank-1 Approximation of M2:")
print(X1_2)
plot_matrix(X1_2)
plt.title("Rank-1 Approximation of M2")

print("Rank-1 Approximation of M3:")
print(X1_3)
plot_matrix(X1_3)
plt.title("Rank-1 Approximation of M3")

print("Rank-1 Approximation of M4:")
print(X1_4)
plot_matrix(X1_4)
plt.title("Rank-1 Approximation of M4")

print("Rank-1 Approximation of M5:")
print(X1_5)
plot_matrix(X1_5)
plt.title("Rank-1 Approximation of M5")

print("Rank-1 Approximation of M6:")
print(X1_6)
plot_matrix(X1_6)
plt.title("Rank-1 Approximation of M6")
```

```
Rank-1 Approximation of M1:
[[1. 1. 1. 0. 0.]
 [1. 1. 1. 0. 0.]
 [1. 1. 1. 0. 0.]
 [0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]]
Rank-1 Approximation of M2:
[[0. 0. 0. 0. 0.]
```

```
 [0. 2. 1. 2. 0.]
 [0. 2. 1. 2. 0.]
 [0. 2. 1. 2. 0.]
 [0. 0. 0. 0. 0.]]
Rank-1 Approximation of M3:
[[0. 0. 0. 0.]
 [0. 1. 1. 1.]
 [0. 1. 1. 1.]
 [0. 1. 1. 1.]
 [0. 1. 1. 1.]]
Rank-1 Approximation of M4:
[[1. 1. 1. 0. 0.]
 [1. 1. 1. 0. 0.]
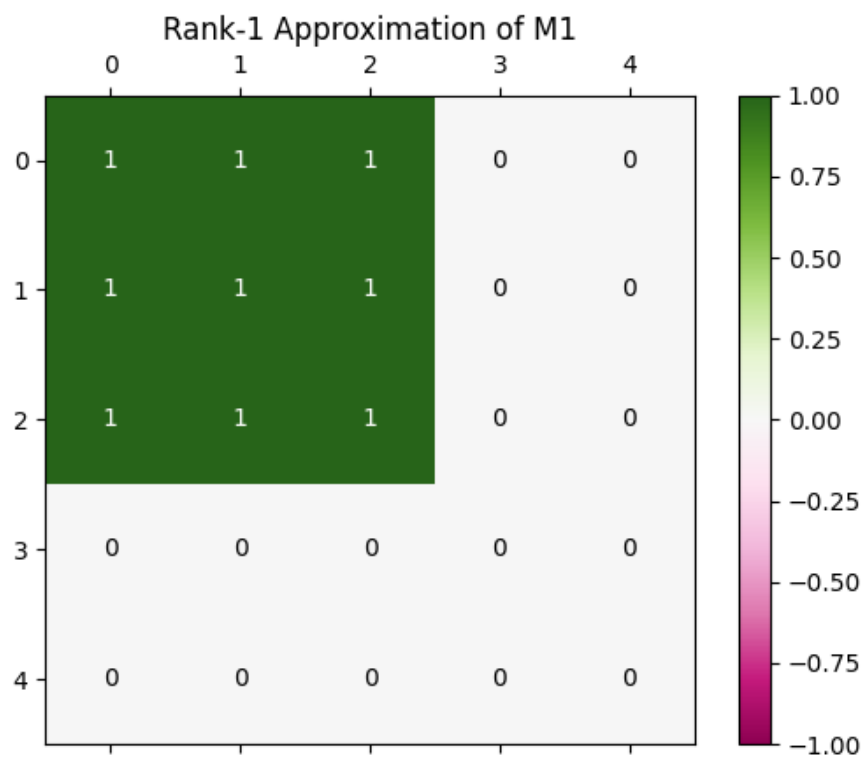 [1. 1. 1. 0. 0.]
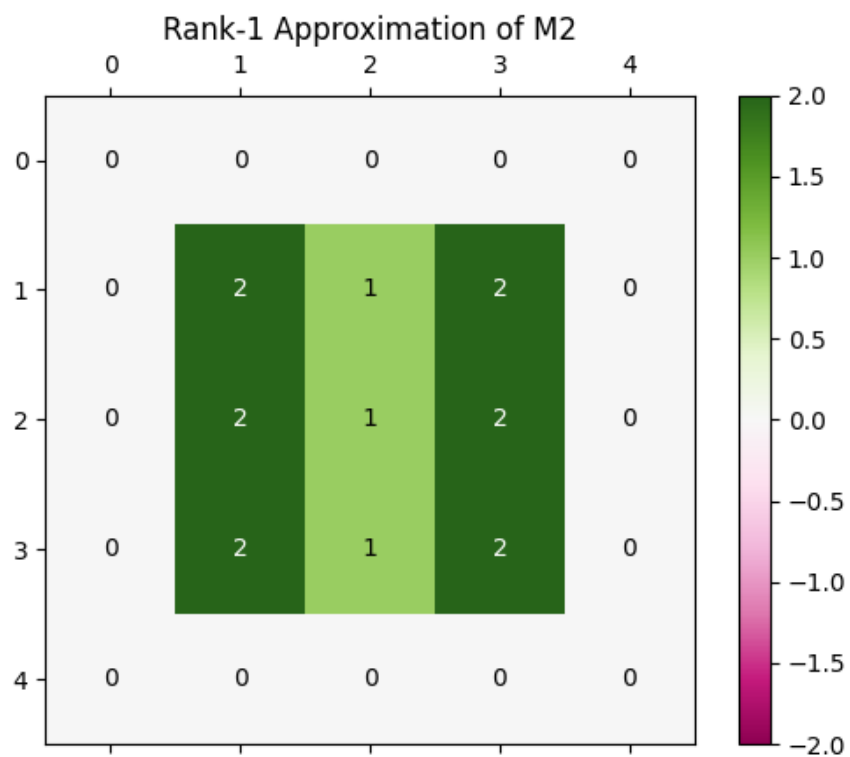 [0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]]
Rank-1 Approximation of M5:
[[0.55316953 0.55316953 0.86380344 0.55316953 0.55316953]
 [0.55316953 0.55316953 0.86380344 0.55316953 0.55316953]
 [0.86380344 0.86380344 1.34887469 0.86380344 0.86380344]
 [0.55316953 0.55316953 0.86380344 0.55316953 0.55316953]
 [0.55316953 0.55316953 0.86380344 0.55316953 0.55316953]]
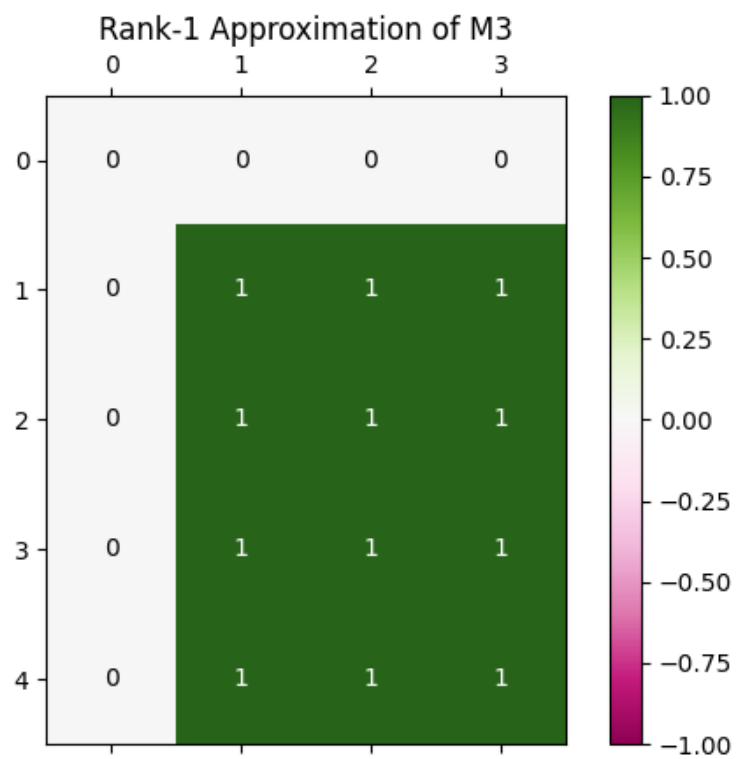Rank-1 Approximation of M6:
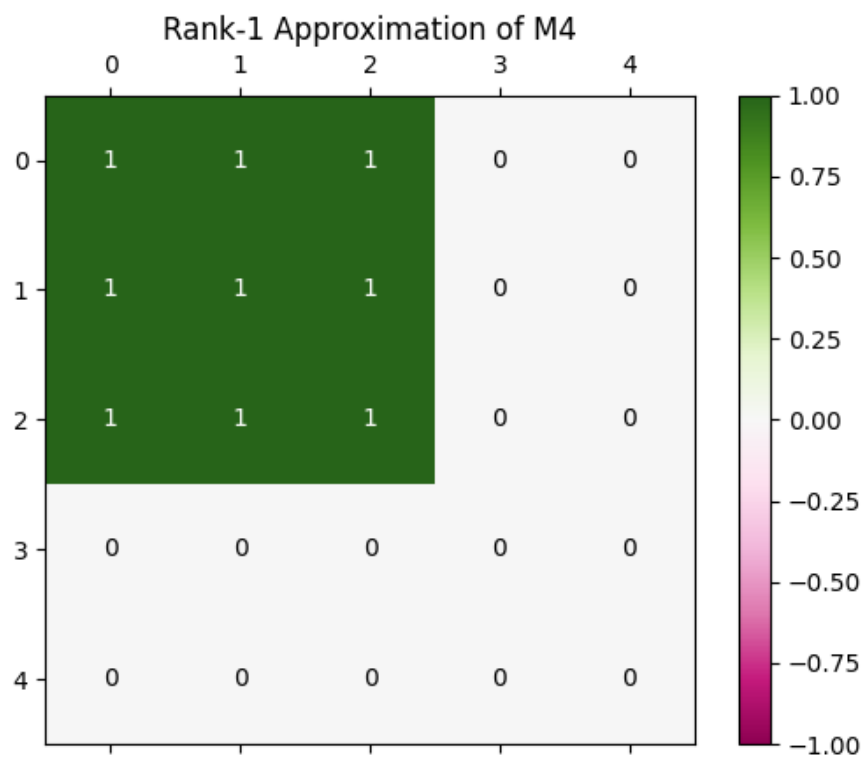[[1.03033009 1.03033009 0.85355339 1.03033009 1.03033009]
 [1.03033009 1.03033009 0.85355339 1.03033009 1.03033009]
 [0.85355339 0.85355339 0.70710678 0.85355339 0.85355339]
 [1.03033009 1.03033009 0.85355339 1.03033009 1.03033009]
 [1.03033009 1.03033009 0.85355339 1.03033009 1.03033009]]
```

[44]: Text(0.5, 1.0, 'Rank-1 Approximation of M6')

Rank-1 Approximation of M1

Rank-1 Approximation of M2

Rank-1 Approximation of M3

Rank-1 Approximation of M4

Rank-1 Approximation of M5

	0	1	2	3	4
0	0.553	0.553	0.864	0.553	0.553
1	0.553	0.553	0.864	0.553	0.553
2	0.864	0.864	1.35	0.864	0.864
3	0.553	0.553	0.864	0.553	0.553
4	0.553	0.553	0.864	0.553	0.553

Rank-1 Approximation of M6

	0	1	2	3	4
0	1.03	1.03	0.854	1.03	1.03
1	1.03	1.03	0.854	1.03	1.03
2	0.854	0.854	0.707	0.854	0.854
3	1.03	1.03	0.854	1.03	1.03
4	1.03	1.03	0.854	1.03	1.03

The rank-1 approximation of a matrix keeps the largest singular value and its corresponding singular vectors, representing the most dominant pattern or structure within the matrix. By reducing each matrix to a rank-1 form, we simplify it while retaining the key features identified through Singular Value Decomposition (SVD). The following evaluations provide an analysis of each matrix's rank-1 approximation, based on the patterns revealed by their singular values and vectors.

Matrix M1: - Original Structure: Matrix has a clear structure with a 3x3 block of ones in the top-left corner, while the rest of the elements are zeros. - Rank-1 Approximation: The best rank-1 approximation reproduces the 3x3 block of ones in the top rows while setting the other entries close to zero. - Largest Singular Value: The largest singular value of M1 is 3, which captures the repeated structure of the ones. - Interpretation: This dominant singular value reflects the sum of these rows, and the corresponding singular vectors emphasize the strong block pattern. The approximation accurately represents the main structure by focusing on the block of ones while ignoring the zero rows.

Matrix M2: - Original Structure: Matrix M2 has a unique pattern with repeated rows of [0, 2, 1, 2, 0], surrounded by zero rows on the top and bottom. - Rank-1 Approximation: The rank-1 approximation preserves the [0, 2, 1, 2, 0] pattern across three rows, reflecting the structure. - Largest Singular Value: Approximately 5.2, this high singular value indicates the strong presence of the pattern in three rows. - Interpretation: The approximation captures the repetition effectively, as the corresponding singular vectors align with the [0, 2, 1, 2, 0] structure. This singular value's magnitude suggests that this row pattern contributes most significantly to the matrix's structure.

16

Matrix M3: - Original Structure: Matrix M3 shows a consistent pattern with four rows of [0, 1, 1, 1], while the first row is entirely zeros. - Rank-1 Approximation: The rank-1 approximation retains this [0, 1, 1, 1] row pattern across all non-zero rows. - Largest Singular Value: Approximately 3.46, this value reflects the dominant structure of four repeated rows. - Interpretation: The approximation captures the clear pattern with minimal distortion, focusing on the repeated row. This pattern, being consistent across multiple rows, allows for a strong representation through the rank-1 form.

Matrix M4: - Original Structure: Matrix M4 contains two distinct blocks: a 3x3 block of ones in the top-left and a 2x2 block of ones in the bottom-right corner. - Rank-1 Approximation: The rank-1 approximation captures primarily the 3x3 block [1, 1, 1, 0, 0], as it has the stronger singular value. - Largest Singular Value: The largest singular value is 3, reflecting the dominance of the first block over the second. - Interpretation: The second block is omitted in the rank-1 approximation due to its lower singular value, indicating it has less influence on the overall structure. Thus, the approximation emphasizes the main block of ones, highlighting its structural prominence.

Matrix M5: - Original Structure: Matrix M5 has a more complex pattern, with a M5 block of ones in the top rows, followed by rows [0, 0, 1, 1, 1]. - Rank-1 Approximation: This matrix's rank-1 approximation captures a blend of the two prominent row patterns but with averaged values to represent both. - Largest Singular Value: Approximately 3.56, this singular value captures the contribution from both block patterns but in a balanced manner. - Interpretation: The approximation highlights the structure of the repeated rows [1, 1, 1, 0, 0] and [0, 0, 1, 1, 1]. Due to their balanced impact, the rank-1 form blends the values, providing a generalized yet representative structure of the overall matrix.

Matrix M6: - Original Structure: Matrix M6 is close to uniform, with nearly all elements set to 1 except for a central zero, giving it a primarily homogeneous appearance. - Rank-1 Approximation: The best rank-1 approximation retains a structure close to an all-ones matrix, with slightly reduced values near the central zero. - Largest Singular Value: Approximately 4.83, this value indicates the matrix's high uniformity. - Interpretation: Since M6 is almost entirely filled with ones, the approximation effectively captures this uniformity, representing it with minor deviations near the central zero. This matrix's homogeneous structure allows the rank-1 form to provide a faithful and intuitive summary.

Across these matrices, the rank-1 approximations consistently capture the most significant patterns in the structures, focusing on the blocks or rows with the highest repetition or uniformity. Each matrix's largest singular value signifies the strength of its primary pattern, while the singular vectors align to express these patterns within the rank-1 approximation. This approach simplifies each matrix effectively, preserving the core structure and providing a concise representation through the SVD decomposition.

2.3 1d

```
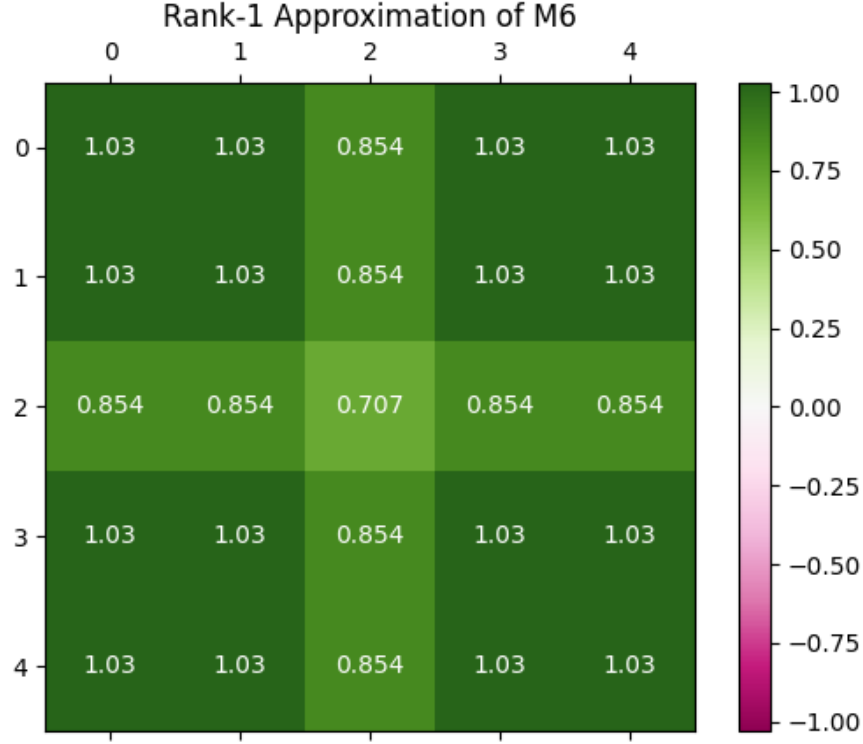[45]: # Count the number of non-zero singular values
      non_zero_singular_values_M6 = np.sum(s6 > 1e-10)
      print(f"Number of non-zero singular values for M6:␣
        ↪{non_zero_singular_values_M6}")
```

```
Number of non-zero singular values for M6: 2
```

```
[46]: # Another method to compute the rank is matrix_rank.
      rank_M6 = np.linalg.matrix_rank(M6)
      print(f"Rank of M6 using matrix_rank: {rank_M6}")
```

Rank of M6 using matrix_rank: 2

Matrix Structure: M_6 is almost entirely filled with ones, with only a single zero in the center. This structure makes it very close to being a rank-1 matrix (like a matrix of all ones), which would normally have only one non-zero singular value. The single zero at the center slightly disrupts this perfect rank-1 structure, introducing just enough variation to increase the rank by 1, making the expected rank 2.

Expected Rank Analysis: If M_6 were a full matrix of ones, it would have rank 1 because all rows and columns are linearly dependent. Adding the single zero disrupts the complete dependency but only slightly, leading us to expect an increase to rank 2. This is because the matrix now has just one dimension of deviation from a perfectly rank-1 structure.

Numerical Results from NumPy: NumPy's output of 2 non-zero singular values and a computed rank of 2 aligns with this reasoning. NumPy correctly identifies that only two singular values are significant enough to count as non-zero (above the threshold set for numerical precision), resulting in a practical rank of 2.

So, the rank of M_6 is 2, and NumPy's results confirm this by reporting two non-zero singular values. The rank calculation from NumPy aligns with the theoretical expectation for this matrix, considering that the single zero only introduces enough variation to increase the rank to 2.

3   2 The SVD on Weather Data

```
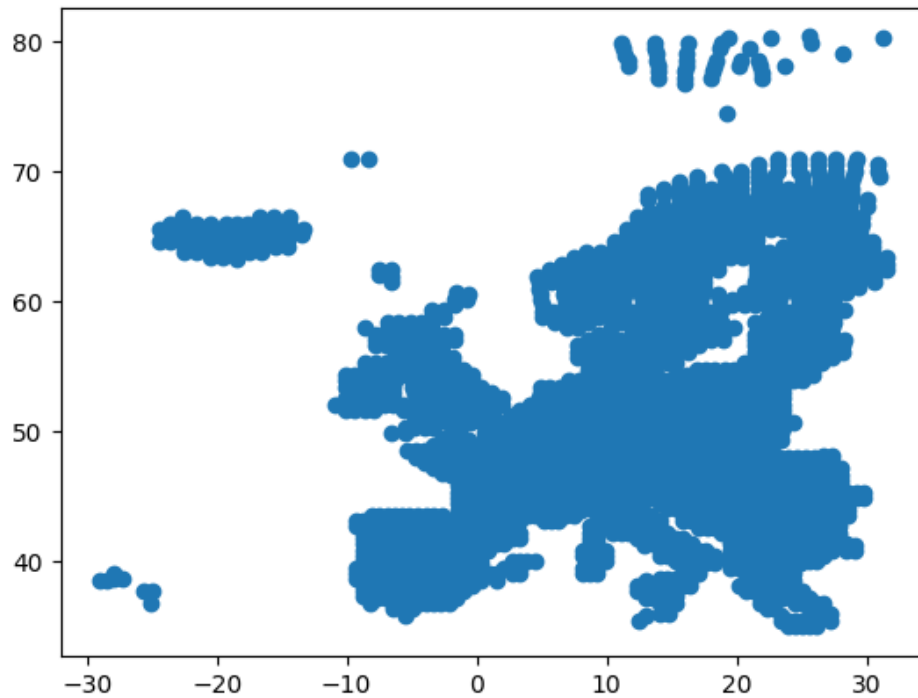[47]: # Load the data
      climate = pd.read_csv("data/worldclim.csv")
      coord = pd.read_csv("data/worldclim_coordinates.csv")
      lon = coord["lon"]
      lat = coord["lat"]
```

```
[48]: # Plot the coordinates
      plot_xy(lon, lat)
```

3.1 2a

```
[49]: # Normalize the data to z-scores. Store the result in X.
      X = (climate - climate.mean()) / climate.std()

      # Check if the assumptions for normalizing the data are reasonable
      if np.allclose(X.mean(), 0, atol=1e-10):
          print("Mean of normalized data is close to 0:")
          print(X.mean())
      if np.allclose(X.std(), 1, atol=1e-10):
          print("Standard deviation of normalized data is close to 1:")
          print(X.std())
```

```
Mean of normalized data is close to 0:
min1    -4.415023e-17
min2    -4.415023e-17
min3     4.415023e-17
min4     2.207511e-17
min5     9.933801e-17
min6    -3.476830e-16
min7     6.622534e-17
```

19

```
min8      1.103756e-17
min9     -1.545258e-16
min10     1.545258e-16
min11    -4.415023e-17
min12     1.324507e-16
max1      0.000000e+00
max2     -4.415023e-17
max3      4.415023e-17
max4     -8.830046e-17
max5     -1.103756e-16
max6     -3.476830e-16
max7      2.980140e-16
max8     -9.933801e-17
max9     -8.830046e-17
max10    -2.649014e-16
max11     2.207511e-16
max12     1.766009e-16
avg1      0.000000e+00
avg2     -4.415023e-17
avg3     -8.830046e-17
avg4     -1.766009e-16
avg5     -2.317887e-16
avg6      1.131350e-16
avg7     -1.766009e-16
avg8      9.050797e-16
avg9      1.103756e-16
avg10     0.000000e+00
avg11     0.000000e+00
avg12    -4.415023e-17
rain1     0.000000e+00
rain2     0.000000e+00
rain3     1.766009e-16
rain4    -8.830046e-17
rain5     2.428263e-16
rain6    -1.490070e-16
rain7    -9.933801e-17
rain8     2.759389e-18
rain9     8.830046e-17
rain10   -2.649014e-16
rain11   -6.622534e-17
rain12   -4.415023e-17
dtype: float64
Standard deviation of normalized data is close to 1:
min1      1.0
min2      1.0
min3      1.0
min4      1.0
min5      1.0
```

```
min6      1.0
min7      1.0
min8      1.0
min9      1.0
min10     1.0
min11     1.0
min12     1.0
max1      1.0
max2      1.0
max3      1.0
max4      1.0
max5      1.0
max6      1.0
max7      1.0
max8      1.0
max9      1.0
max10     1.0
max11     1.0
max12     1.0
avg1      1.0
avg2      1.0
avg3      1.0
avg4      1.0
avg5      1.0
avg6      1.0
avg7      1.0
avg8      1.0
avg9      1.0
avg10     1.0
avg11     1.0
avg12     1.0
rain1     1.0
rain2     1.0
rain3     1.0
rain4     1.0
rain5     1.0
rain6     1.0
rain7     1.0
rain8     1.0
rain9     1.0
rain10    1.0
rain11    1.0
rain12    1.0
dtype: float64
```

Considering the data we are using, the assumptions for normalizing the data via the z-score are reasonable. Here's why:

1. **Nature of the Data**: The dataset contains climatic variables (minimum, maximum, and

average temperatures, as well as average rainfall) across different regions in Europe. These variables are continuous and typically follow a normal distribution, making them suitable for z-score normalization.

2. **Scale and Units**: The climatic variables are measured in different units (e.g., temperature in degrees Celsius, rainfall in millimeters). Z-score normalization standardizes these variables to a common scale with a mean of 0 and a standard deviation of 1, which is essential for ensuring that each variable contributes equally to the analysis.

3. **Comparability**: Normalizing the data via z-score allows for comparability across different climatic variables and regions. This is particularly important for data mining and matrix-based analyses, where the relative differences between variables need to be preserved.

4. **Statistical Properties**: The z-score normalization assumes that the data is approximately normally distributed. Given the large number of observations (2575 rows), the Central Limit Theorem suggests that the distribution of sample means will be approximately normal, justifying the use of z-score normalization.

5. **Verification**: The code provided checks if the mean of the normalized data is close to 0 and the standard deviation is close to 1, confirming that the z-score normalization has been applied correctly.

6. **Supporting Literature**: Additionally, we just want to reference this paper, which suggests preprocessing the climate data by normalizing it as an example of typical procedure in working with climate data: Deepak Sharma, Priti Sharma in November 2023.

In summary, given the continuous nature of the climatic variables, the need for comparability, and the statistical properties of the data, the assumptions for normalizing the data via the z-score are reasonable and appropriate for this dataset.

```
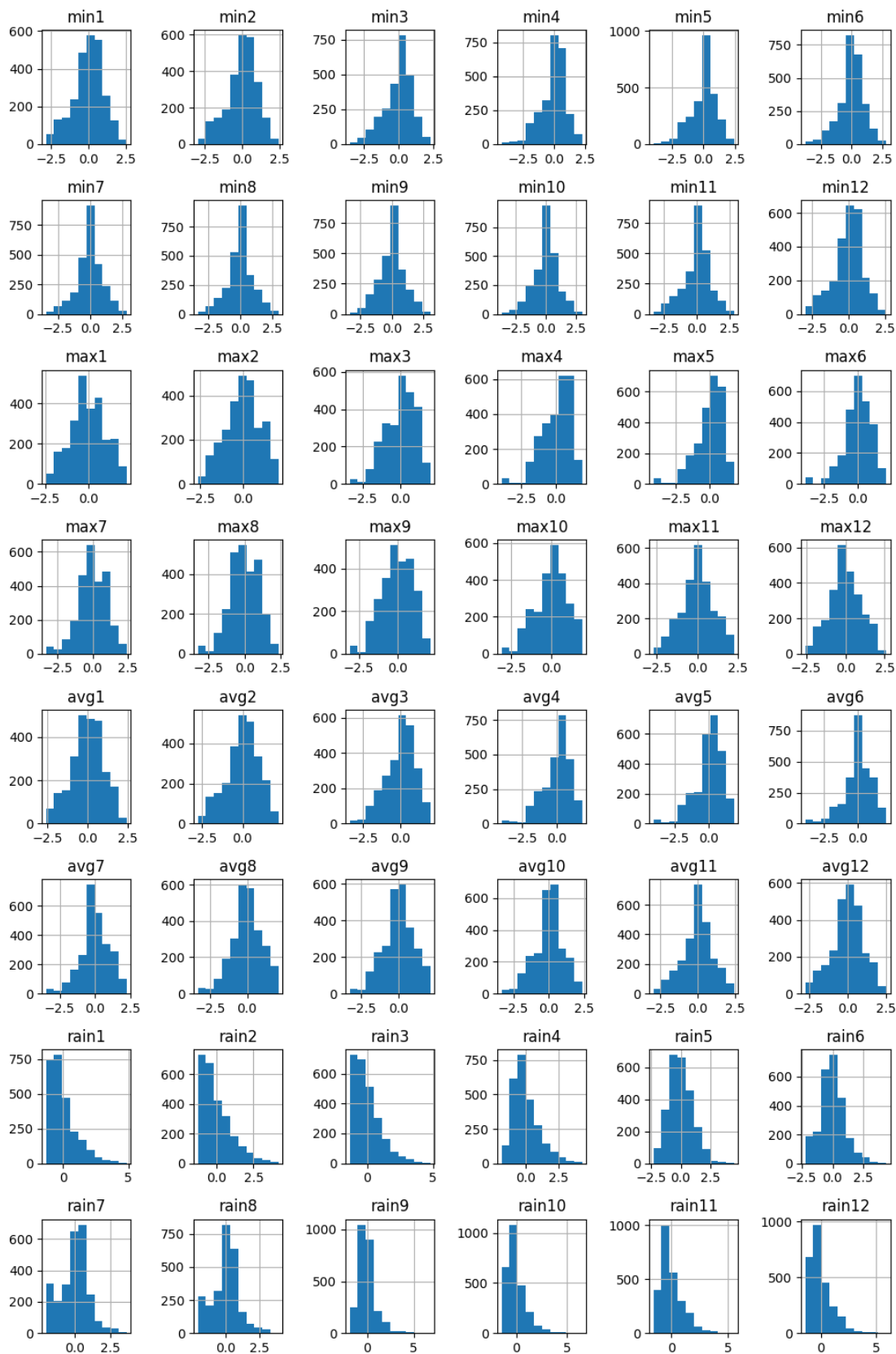[50]: # Plot histograms of attributes
      X.hist(layout=(8, 6), figsize=(10, 15))
      plt.tight_layout()
      plt.show()
```

3.1.1 Analysis of Normalized Climate Data Histogram

General Observations

- The data appears to be normalized overall.
- Most features show a distribution centered around 0, which is typical for normalized data.

Specific Observations on Rain Features (1 to 12)

- The rain features exhibit a left-skewed distribution.
- High values are concentrated around 0 and to the left of it.
- The distribution slowly decreases to the right, covering a wider range on the x-axis with lower heights.
- This does not follow the typical bell curve of a normal distribution.

Conclusion

- The normalization process seems to have worked for most features.
- The rain features' left-skewed distribution might indicate an inherent characteristic of the data or a need for further preprocessing.

3.2 2b

```python
[51]:  # Compute the SVD of the normalized climate data and store it in variables
       # ↪U,s,Vt.  What
       # is the rank of the data?

       # Compute the SVD of the normalized climate data
       U, s, Vt = svd(X, full_matrices=False)

       # Compute the rank of the data
       rank_X = np.sum(s > 1e-10)

       print(f"Rank of the normalized climate data: {rank_X}")
       # Print the first few singular values to understand the variance captured
       print("First 10 singular values of the normalized climate data:")
       print(s[:10])

       # Plot the singular values to visualize the variance captured by each component
       plt.figure(figsize=(10, 6))
       plt.plot(s, marker='o')
       plt.title('Singular Values of the Normalized Climate Data')
       plt.xlabel('Index')
       plt.ylabel('Singular Value')
       plt.grid(True)
       plt.show()
```

```
Rank of the normalized climate data: 48
First 10 singular values of the normalized climate data:
[290.16602997 150.63956473  88.47645549  59.17449467  52.11009177
  27.45679147  22.12984534  15.23769162  14.1294369   12.02665177]
```


Singular Values of the Normalized Climate Data
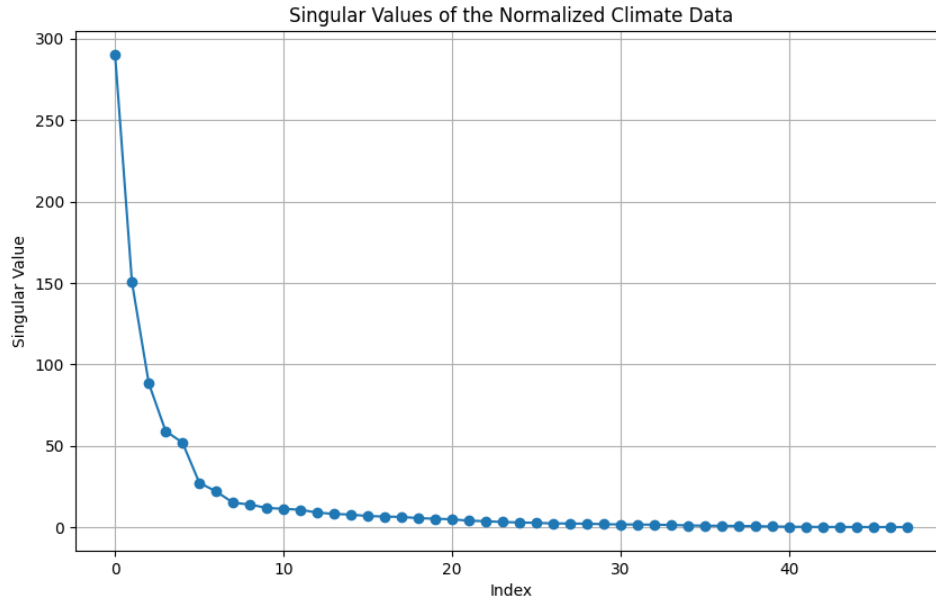
3.2.1 Evaluation:

1. **Rank of the Data**:
   - The rank of the normalized climate data is 48, which indicates that the data matrix has full rank. This means that all 48 dimensions (features) are linearly independent and contribute to the variance in the data.
2. **Singular Values**:
   - The first singular value (290.16602997) is significantly larger than the others, indicating that the first principal component captures the most variance in the data.
   - The subsequent singular values decrease in magnitude, showing the diminishing contribution of each additional principal component to the total variance.
   - The first 10 singular values capture a substantial amount of the total variance, with the first few singular values being particularly dominant.

The SVD results suggest that the normalized climate data has a complex structure with multiple significant components. The full rank of 48 implies that all features are important for capturing the variability in the data. The dominance of the first few singular values indicates that a few principal components can explain a large portion of the variance, which is typical in high-dimensional datasets.

3.3 2c

The goal of the visualization is to depict the spatial distribution of the values in the first five columns of the left singular vector matrix (U). Each plot uses the longitude and latitude of each data point as the x and y coordinates, respectively, and the corresponding entry in the left singular vector to color each point. This helps in understanding the geographical patterns and variations captured by the singular vectors.

```python
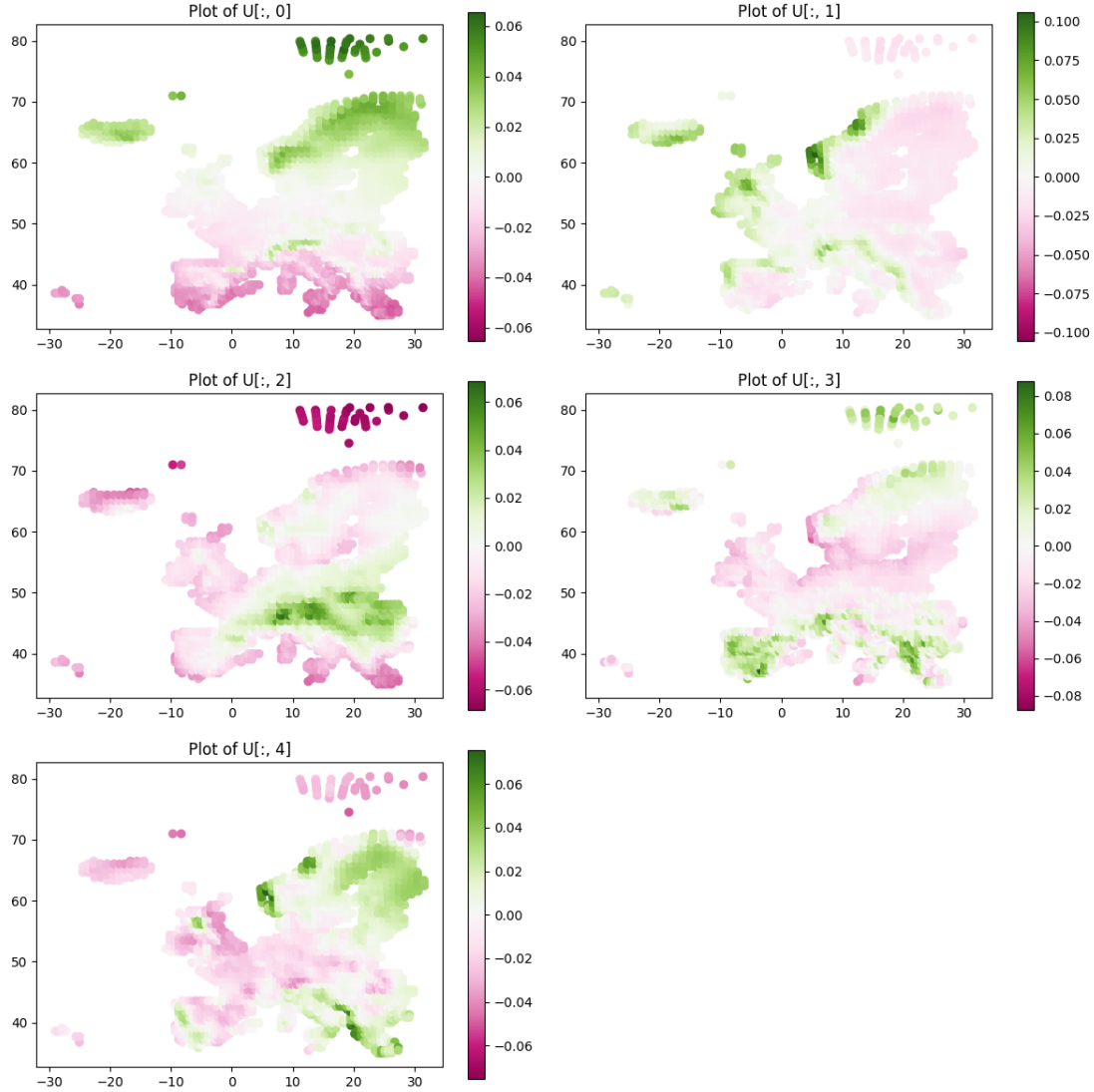# Here is an example plot.
#plot_xy(lon, lat, U[:, 0])

fig, axs = plt.subplots(3, 2, figsize=(12, 12))
axs = axs.flatten()

for i in range(5):
    plot_xy(lon, lat, U[:, i], axis=axs[i])
    axs[i].set_title(f"Plot of U[:, {i}]")
    # Remove the last axis if i == 4
    if i == 4:
        fig.delaxes(axs[-1])

plt.tight_layout()
plt.show()
```

3.3.1 Analysis of Left Singular Vector Plots

General Interpretation of Singular Vectors   Each of these plots represents one of the first five singular vectors, capturing unique patterns in the dataset. The geographical coordinates (longitude and latitude) define the position of each data point, while the color scale (from pink to green) represents the intensity of each entry in the singular vector. Regions colored in green have high values, while regions in pink have low values for the given singular vector component.

Since the singular values in this dataset suggest that the first component captures a dominant portion of the variance, followed by diminishing contributions in subsequent components, we should expect the initial singular vector plots to display broad, high-variance spatial patterns, with finer variations appearing as we progress to higher-order vectors.

Plot of $U[:, 0]$ (First Singular Vector) - Description: The plot of $U[:, 0]$ shows a strong contrast, with

27

the top (north) part of the map displaying a dark green concentration, while the middle and lower areas exhibit light green to pink shades. - Interpretation: The first singular vector captures a major climatic gradient likely associated with latitude. The green regions, possibly the northern areas, suggest distinct climate conditions (e.g., lower temperatures or different rainfall patterns) compared to more southern regions. This vector might represent a general north-south climatic gradient in Europe, consistent with the continent's latitudinal temperature and precipitation distribution. - Key Observation: This vector is likely dominated by temperature and seasonal variations from north to south, with a noticeable contrast that implies a primary variance direction in the climate dataset.

Plot of $U[:, 1]$ (Second Singular Vector) - Description: The plot of $U[:, 1]$ reveals a more diffused pattern compared to $U[:, 0]$, with light green in the center and pinkish regions along the edges. - Interpretation: This second singular vector seems to capture a secondary pattern, possibly related to continentality (inland vs. coastal differences) or another climate gradient that differs from the primary north-south trend. The lighter central area could represent regions with moderate climate, while the edges might capture more extreme or specific local climates. - Key Observation: This component adds additional structure, potentially reflecting subtler climatic influences such as proximity to coastlines, altitude, or distance from the ocean.

Plot of $U[:, 2]$ (Third Singular Vector) - Description: The third plot has a highly contrasting color scheme with noticeable pink patches at the top and bottom, and a stronger green band in the central region. - Interpretation: The third singular vector might highlight localized climate characteristics, potentially capturing regional climate differences within Europe that are not purely north-south or inland-coastal. This distribution suggests a third climatic factor, perhaps linked to specific seasonal patterns or temperature extremes in certain geographic locations. - Key Observation: The alternation between green and pink regions indicates that this vector likely captures seasonal or altitude-driven differences, where certain regions differ markedly from neighboring areas in their climate variability.

Plot of $U[:, 3]$ (Fourth Singular Vector) - Description: The fourth plot shows an even more nuanced pattern, with smaller green and pink patches scattered throughout. The coloring is more granular and less uniform, with both green and pink spots distributed in clusters. - Interpretation: This singular vector might capture highly localized climate features or micro-climatic zones, like the north of Scandinavia and the Mediterranean. The scattered patches imply regional variability that could stem from local factors such as mountain ranges, bodies of water, or unique land formations that affect climate independently of broad regional patterns. - Key Observation: The presence of scattered clusters in both colors points to fine-scale variations, suggesting this component picks up on minor climatic influences that impact specific areas rather than broad regional zones.

Plot of $U[:, 4]$ (Fifth Singular Vector) - Description: The fifth plot also displays a mix of green and pink, but with smaller clusters and a more balanced distribution of colors across the map. We can interpret it as a east-west-component. - Interpretation: The fifth singular vector likely represents even finer-scale variations or residual climatic differences that aren't captured by the primary vectors, specifically focusing on eastern - western climate differences. This could include isolated anomalies or unique micro-climates. - Key Observation: This vector has the smallest spatial coherence, implying it accounts for the smallest amount of variance and may capture irregularities, outlier climatic patterns, or noise in the data.

Comparative Summary and Interpretation - Variance Explained: As we move from $U[:, 0]$ to $U[:, 4]$, the singular vectors capture progressively finer and more localized patterns of climate variability.

The first vector represents a dominant regional trend (probably north-south climate gradient), while subsequent vectors add complexity, capturing coastal effects, regional anomalies, and local climatic peculiarities. - Pattern Complexity: The gradual shift from large, consistent regions of color in $U[:,0]$ to scattered patches in $U[:,4]$ aligns with the decreasing singular values, reflecting a hierarchy in the strength and scale of these climatic patterns. - Climatic Insight: The primary vectors $U[:,0]$ and $U[:,1]$ suggest broad, structural climatic trends, likely driven by latitude and geographic location within Europe. The higher-order vectors, from $U[:,2]$ onward, capture more nuanced or region-specific details, which may include unique local climates or anomalies not explained by major trends.

This visual exploration reveals that the dataset's climate variability is highly structured, with major geographical and seasonal trends captured by the initial singular vectors, and finer, region-specific climatic features uncovered by the higher-order components. This approach illustrates how SVD can decompose complex, high-dimensional datasets into interpretable spatial and climatic patterns, capturing the hierarchy of climate variability across Europe.

```python
# Print the U, S, and Vt matrices
print("U matrix:")
print(U)

print("\nSingular values (S):")
print(s)

print("\nVt matrix:")
print(Vt)

# Visualize the U and Vt matrices in a single plot with subfigures
fig, axs = plt.subplots(2, 1, figsize=(10, 12))

# U matrix
axs[0].imshow(U, aspect='auto', cmap='viridis')
axs[0].set_title('U Matrix')
axs[0].set_xlabel('Components')
axs[0].set_ylabel('Samples')
fig.colorbar(axs[0].images[0], ax=axs[0])

# Vt matrix
axs[1].imshow(Vt, aspect='auto', cmap='viridis')
axs[1].set_title('Vt Matrix')
axs[1].set_xlabel('Features')
axs[1].set_ylabel('Components')
fig.colorbar(axs[1].images[0], ax=axs[1])

plt.tight_layout()
plt.show()
```

```
U matrix:
[[-0.02985651  0.03023726 -0.02532763 …  0.01131257 -0.00498623
  -0.0111336 ]
```

```
[-0.0283224   0.02994646 -0.02256971 …  0.01278342 -0.01337286
   0.01395358]
 [-0.03072081   0.03133108 -0.02705535 … -0.01491217   0.02347298
  -0.00982949]
 …
 [ 0.02845906 -0.00982454 -0.03288548 …  0.01323813   0.00529513
   0.03243245]
 [ 0.03147069 -0.01222644 -0.02647448 … -0.03725883   0.0153384
  -0.01561221]
 [ 0.02901476 -0.00533857 -0.03742868 …  0.00376358 -0.00236453
  -0.03357857]]

Singular values (S):
[2.90166030e+02 1.50639565e+02 8.84764555e+01 5.91744947e+01
 5.21100918e+01 2.74567915e+01 2.21298453e+01 1.52376916e+01
 1.41294369e+01 1.20266518e+01 1.14744759e+01 1.09188627e+01
 9.14526380e+00 8.39529310e+00 7.93057599e+00 7.06637363e+00
 6.74109591e+00 6.51712004e+00 5.76693636e+00 5.39573839e+00
 5.06780457e+00 4.20956360e+00 3.88432125e+00 3.37927249e+00
 3.11950833e+00 2.88128642e+00 2.53296890e+00 2.48117703e+00
 2.32922526e+00 2.07690010e+00 1.90511665e+00 1.86356099e+00
 1.72296647e+00 1.60127352e+00 1.28311464e+00 1.12585686e+00
 1.04938033e+00 9.84336239e-01 8.39757277e-01 6.56575889e-01
 4.93935950e-01 4.13256214e-01 3.78169169e-01 3.47174687e-01
 3.20825013e-01 3.06144826e-01 3.00678189e-01 2.55249267e-01]

Vt matrix:
[[-1.58274873e-01 -1.61805700e-01 -1.67211061e-01 … -2.35436083e-03
  -3.39660239e-02 -3.72464665e-02]
 [ 8.90582198e-02  8.04359572e-02  5.03430022e-02 …  3.12061021e-01
   3.06472191e-01  3.06633544e-01]
 [-1.17233954e-01 -8.75096875e-02 -3.64678893e-02 … -6.56222617e-02
  -5.09478746e-02 -8.61838473e-02]
 …
 [ 5.48447100e-02  1.06187345e-01 -1.90508111e-01 …  1.25514912e-03
   4.95443370e-04  4.22241735e-04]
 [ 2.62380562e-02 -2.17170586e-02 -1.80741872e-02 … -9.13291691e-04
   1.51397958e-03  6.38416250e-04]
 [ 2.71071823e-01 -2.82200563e-01  1.03236964e-01 … -2.90032056e-04
  -4.85381867e-04 -2.95980957e-04]]
```
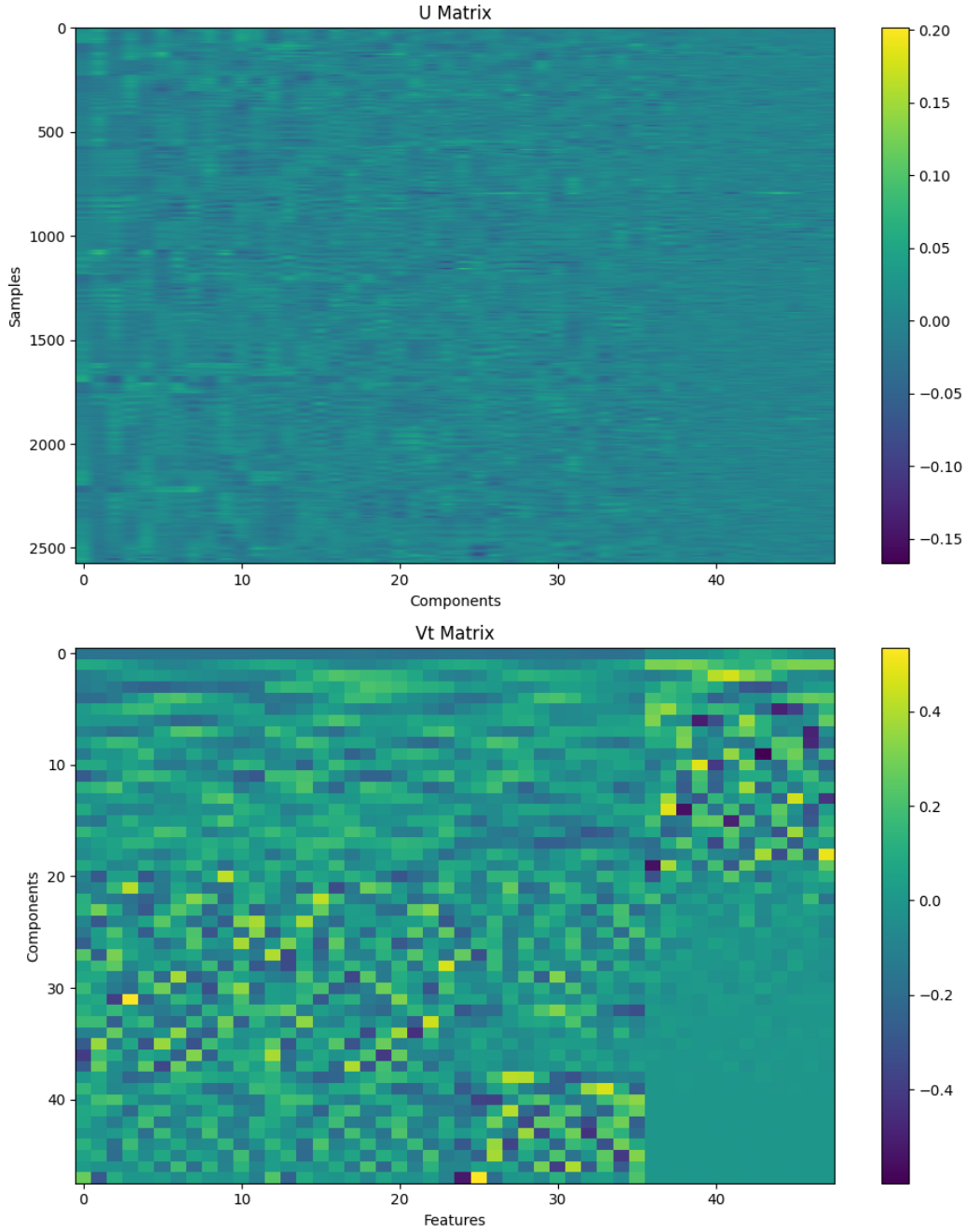
3.3.2 Analysis of U and V^T Matrices from SVD Decomposition

In this visualization, we examine the U and V^T matrices obtained from the Singular Value Decomposition (SVD) of the WorldClim dataset. These matrices represent essential components of the data's underlying structure and can offer insights into the patterns and relationships between the samples and features. - U Matrix: Each column in U corresponds to a left singular vector, associated with a particular component of the climate data's variance. Rows represent samples

(data points with specific latitude and longitude coordinates). - V^T Matrix: Each row in V^T corresponds to a right singular vector, which maps to a feature component (columns in the original data matrix). The entries within these vectors indicate how the original features contribute to each principal component.

This analysis will explore both matrices and draw comparisons to understand the relationships between components, samples, and features within the dataset.

Observations on the U Matrix Plot

1. Distribution of Values:

- The color range in the U matrix plot, from dark purple (negative) to bright yellow (positive), represents the distribution of values for each component across all samples.
- The coloring is generally consistent across the majority of the components, though subtle variations appear, indicating that most components have balanced contributions across samples with few extremes. This balance suggests that no single component overwhelmingly dominates any particular geographical area, aligning with the geographic diversity captured in climate data.

2. Sample Variability:

- The intensity of colors along the rows (representing samples) varies, indicating that different regions contribute differently to each component.
- Some rows are slightly brighter, which might correspond to specific geographical areas with unique climate characteristics that stand out in certain components.

3. Component Patterns:

- The structure is relatively uniform across most components, without any sudden shifts in value. This observation suggests that while there is a primary climatic variance captured by the first few components, no single component dominates the entire dataset.
- The gradual color shifts across the columns hint at subtle changes in climate variation as represented by the components.

Observations on the V^T Matrix Plot

1. Feature Concentration in Components:

- The V^T matrix shows a more complex structure compared to U, with clusters of features concentrated in specific components.
- The concentration of brighter (yellow) or darker (purple) cells in certain rows reflects how some features contribute strongly to specific components while being less relevant in others.
- Notably, the presence of distinct blocks with intense colors suggests that particular groups of features exhibit strong interdependence within certain components, implying correlations between these climate variables (e.g., temperature or precipitation measurements for consecutive months).

2. Component-Feature Interaction:

- The feature values appear more clustered toward the left side of the matrix, indicating that the earlier components explain a larger portion of the variance in the climate data.

- This clustering supports the interpretation that the first few components capture the primary climatic trends, while the later components address finer details or residuals in the data.

3. Granularity and Noise in Higher Components:

- Toward the lower part of the V^T matrix, color patterns appear more scattered, and some rows show a more fragmented color distribution.
- This granularity suggests that these later components capture residual variability or noise, potentially highlighting features with limited or highly localized variance that were not well captured in the main components.

Comparison Between U and V^T Matrices

1. Dimensional Representation:

- The U matrix captures sample-level patterns, while the V^T matrix provides feature-level patterns. Together, they represent how samples and features interact across principal components.
- The consistent patterns in U reflect that the samples (locations) exhibit a stable climate gradient, while the more complex structure in   highlights that certain groups of features (e.g., monthly temperatures or rainfall) are heavily correlated within specific components.

2. Interpretation of Geographic vs. Climatic Components:

- The U matrix, with relatively uniform values across components, suggests that the primary variation in the data has a broad geographic effect, likely correlating with the major climatic gradients (such as latitude or altitude).
- In contrast, the V^T matrix shows that climate features exhibit more specific correlations, which might correspond to seasonal patterns or specific environmental factors (e.g., temperature variation in winter vs. summer months).

3. Primary vs. Secondary Variance:

- The U matrix primarily highlights broad trends across samples, aligning with primary climate patterns that impact large regions. This is evident in the smooth transitions along most of the components.
- Meanwhile, V^T captures more detailed feature-level relationships, likely reflecting interdependencies within the data that might be due to seasonal or monthly variations in the climate variables. The structured blocks within V^T suggest that certain climate measurements are strongly tied to each other.

4. Information Hierarchy:

- Both matrices reinforce the idea of a hierarchy in the data. The early components in both U and V^T display smoother, more significant patterns, while later components become more granular and potentially capture minor or residual variability.

The visualizations of U and V^T provide complementary insights: - Broad Climate Patterns: The first few singular vectors represent the primary sources of climate variation across Europe, such as temperature and precipitation gradients likely driven by latitude, altitude, and other regional factors. - Localized Climate Features: Higher-order components capture fine-grained, localized climatic variations, representing specific seasonal dependencies or residual patterns. - Structured

Relationships: The block patterns in V^T suggest that climate variables are not independent; instead, they form clusters that correlate strongly within certain components, capturing seasonal or monthly effects.

This SVD decomposition allows for a layered understanding of the dataset, where dominant components provide a simplified, yet comprehensive view of Europe's climate variability, while higher components offer nuanced insights into specific features and locations. This multiscale representation is valuable for climate analysis, providing both an overarching view and the ability to zoom into finer details.

3.4   2d

```
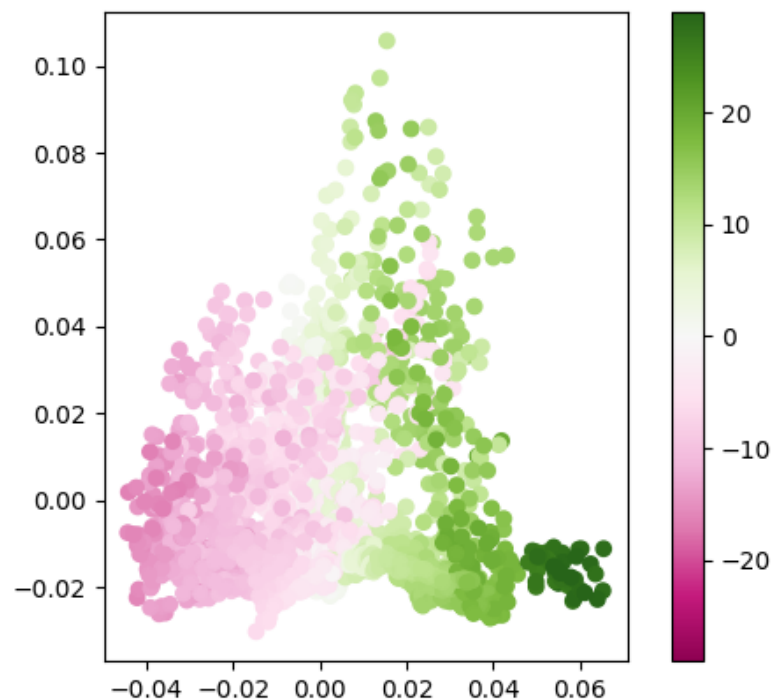[54]: # Here is an example.
      plot_xy(U[:, 0], U[:, 1], lat - np.mean(lat))
```


Interpretation of Scatter Plot Results   The scatter plot visualizes the relationship between columns of matrix U, where each data point is colored according to its North-South or East-West geographical location. By examining the plot, we can draw several important conclusions about the spatial and climatic variability in the dataset, which is related to temperature and rainfall patterns across Europe.

Observations: 1. Concentration of Data Points: - A notable feature is the concentration of data

points around specific y-values (0 and -0.02). This suggests that these points share a common characteristic or climatic feature. The clustering could indicate a uniform climatic pattern across a certain geographical area, perhaps a specific climate zone or region with similar temperature and rainfall characteristics. 2. Pink Dots: - The pink dots are predominantly located on the left side of the plot. This indicates that the left side corresponds to a specific climatic trend or geographical region that is reflected in the pink color. The increasing intensity of the pink as you move leftward could imply a gradient or more specific relationship to latitude (North-South direction) or climatic conditions (colder or drier climates). - This pattern might be related to higher latitudes, where colder and drier conditions are observed, such as the northern parts of Europe or higher altitudes. 3. Green Dots: - The green dots, on the other hand, are more dispersed along the vertical axis, with values reaching up to 0.1. The spread suggests a greater variability in this region, indicating that this cluster might represent areas with more diverse climatic conditions. - The concentration of dark green dots around the bottom left of the plot, specifically around y-values of -0.02, suggests a denser concentration of data points with similar climatic characteristics in specific regions, likely those with temperate climates or locations that are relatively close to bodies of water.

Analysis:

- Spatial Distribution:
  - The spatial distribution of pink and green dots points to a clear geographical influence on the climate data. Pink dots may represent colder, more northern regions (i.e., north of Europe), while the green dots seem to represent regions with a broader climatic diversity—perhaps areas in the central and southern parts of Europe, or coastal regions with more temperate climates.
  - The fact that the green dots are more spread out vertically and horizontally suggests that these regions experience a wider range of climatic conditions, possibly due to factors like proximity to the coast, elevation, or the diverse latitudes and longitudes within southern Europe.
- Interaction with Previous Results:
  - The scatter plot aligns with our earlier understanding of the singular vectors. For instance, the concentration of pink dots on the left could correspond with the primary climatic gradient captured by the first singular vector, which likely represents the dominant North-South climate gradient (colder north versus warmer south).
  - Similarly, the more dispersed green dots in the vertical direction may relate to the second or third singular vectors, which capture finer regional or altitude-based climatic differences, such as coastal versus inland areas, or the effects of geographical features like mountains or seas.
- Implications for Further Analysis:
  - This clustering and dispersion provide valuable insights into the geography of climate. The pink and green color patterns suggest that climate variability in Europe is influenced by broad regional trends (e.g., latitude) and finer local effects (e.g., altitude, proximity to the ocean).
  - The plot's clustering effect can also guide more targeted analyses of climate regions. For example, regions dominated by pink could be explored for their specific cold or dry climate traits, while regions with green might warrant further investigation into coastal or temperate zone features.

The scatter plot highlights the significant geographical and climatic patterns within the dataset, emphasizing how the North-South or East-West location of data points plays a crucial role in shap-

ing the climate variables. The clustering of pink dots on the left side suggests a strong correlation with colder climates, while the dispersion of green dots reflects more variable climatic conditions, likely in regions further south or near coastlines. This visualization serves as a powerful tool for understanding the underlying structure of climate variability across Europe and offers insights into potential regions of interest for deeper climate studies.

3.5   2e

3.5.1   Methods for Determining the Number of Principal Components

In this section, we explore various methods to determine the optimal number of principal components to retain in our analysis. These methods include:

(i) **Guttman–Kaiser criterion**: This method suggests retaining all principal components with eigenvalues greater than 1. Mathematically, if (_i) represents the eigenvalues of the covariance matrix, then retain components where (_i > 1).

(ii) **90% of squared Frobenius norm**: This method retains enough principal components such that the sum of their squared singular values accounts for at least 90% of the total squared Frobenius norm of the original matrix. Mathematically, if (_i) are the singular values, find the smallest (k) such that ({i=1}^k _i^2 0.9 {i=1}^n _i^2).

(iii) **Scree test**: This method involves plotting the eigenvalues in descending order and looking for an "elbow" point where the rate of decrease sharply slows. The number of components to retain is determined by the position of this elbow.

(iv) **Entropy-based method**: This method uses entropy to measure the amount of information retained by the principal components. The goal is to retain components that maximize the entropy of the retained information. Note that there is no single definition for this method, and it can vary based on the specific approach used. The method we implemented is based on the approach described in the paper by (Monami Banerjee, Nikhil R. Pal (2014))

(v) **Random flipping of signs**: This method involves randomly flipping the signs of the singular values or eigenvalues and observing the effect on the retained components. The goal is to ensure robustness and stability of the selected components. This method is based on the approach described in the paper by (R. Bra, E. Acar, and T. Kolda (2007)).

```
[55]:  # 2e(i) Guttman-Kaiser

       # Guttman-Kaiser criterion: Keep components with eigenvalues (singular values␣
        ↪squared) greater than the average eigenvalue
       average_eigenvalue = np.mean(s**2) # equal to the mean of the singular values␣
        ↪squared
       print (f"Average eigenvalue: {average_eigenvalue}")
       guttman_kaiser_rank = np.sum(s**2 > average_eigenvalue)
       print(f"Guttman-Kaiser criterion suggests rank: {guttman_kaiser_rank}")
```

```
Average eigenvalue: 2573.9999999999973
Guttman-Kaiser criterion suggests rank: 5
```

3.5.2 Evaluation of the Guttman-Kaiser Criterion

Average Eigenvalue: 2573.9999999999973

Guttman-Kaiser Criterion Suggested Rank: 5

Advantages:

1. **Simplicity**: The Guttman-Kaiser criterion is straightforward to apply. It involves calculating the average eigenvalue and retaining components with eigenvalues greater than this average.
2. **Automatic Threshold**: This method provides an automatic threshold for determining the number of components to retain, which can be useful when there is no clear elbow in the scree plot.
3. **Interpretability**: The criterion is easy to interpret and understand, making it accessible for practitioners who may not have a deep statistical background.

Disadvantages:

1. **Overestimation**: The Guttman-Kaiser criterion can sometimes overestimate the number of components to retain, especially in high-dimensional datasets where many components have eigenvalues slightly above the average.
2. **Context-Dependent**: The effectiveness of the criterion can vary depending on the context and the specific characteristics of the dataset. It may not always align with the underlying structure of the data.
3. **Lack of Flexibility**: The criterion does not allow for flexibility or adjustment based on domain knowledge or specific research questions. It applies a fixed rule that may not be optimal for all situations.
4. **Sensitivity to Scale**: The criterion can be sensitive to the scale of the data, which may affect the calculation of eigenvalues and the resulting rank determination.

In summary, while the Guttman-Kaiser criterion is a useful and simple method for determining the number of components to retain, it has limitations that should be considered. It is important to complement this criterion with other methods and domain knowledge to make a well-informed decision about the appropriate rank for the data.

```
[56]:   # 2e(ii) 90% squared Frobenius norm

        # Calculate the cumulative sum of the squared singular values
        cumulative_squared_singular_values = np.cumsum(s**2) # already sorted in␣
        ↪descending order

        # Calculate the total sum of the squared singular values
        total_squared_singular_values = cumulative_squared_singular_values[-1]
        print(f"Total squared Frobenius norm: {total_squared_singular_values}")

        # Find the number of singular values needed to reach 90% of the total squared␣
        ↪Frobenius norm
        rank_90_percent = np.searchsorted(cumulative_squared_singular_values, 0.9 *␣
        ↪total_squared_singular_values) + 1
```

```
print(f"90% of squared Frobenius norm suggests rank: {rank_90_percent}")
```

```
Total squared Frobenius norm: 123551.99999999988
90% of squared Frobenius norm suggests rank: 3
```

3.5.3 Evaluation of the 90% Squared Frobenius Norm Output

Total Squared Frobenius Norm: 123551.99999999988

90% of Squared Frobenius Norm Suggests Rank: 3

Advantages:

1. **Variance Explanation**: This method ensures that the selected components capture a significant portion (90%) of the total variance in the data, providing a comprehensive representation of the dataset.
2. **Data Reduction**: By focusing on the most significant components, this method effectively reduces the dimensionality of the data, simplifying analysis and visualization while retaining most of the important information.
3. **Objective Threshold**: The 90% threshold is a clear and objective criterion, making it easy to apply and interpret without requiring subjective judgment.

Disadvantages:

1. **Arbitrary Threshold**: The choice of 90% is somewhat arbitrary and may not be optimal for all datasets. Different datasets might require different thresholds to balance between variance captured and dimensionality reduction.
2. **Potential Overfitting**: In some cases, retaining too many components to meet the 90% threshold might lead to overfitting, especially if the data contains noise or irrelevant features.
3. **Loss of Minor Variance**: While capturing 90% of the variance, this method might overlook components that capture important but less dominant patterns in the data, potentially missing subtle but meaningful variations.
4. **Computational Complexity**: Calculating the cumulative variance and determining the rank can be computationally intensive for very large datasets, potentially limiting the method's scalability.

```
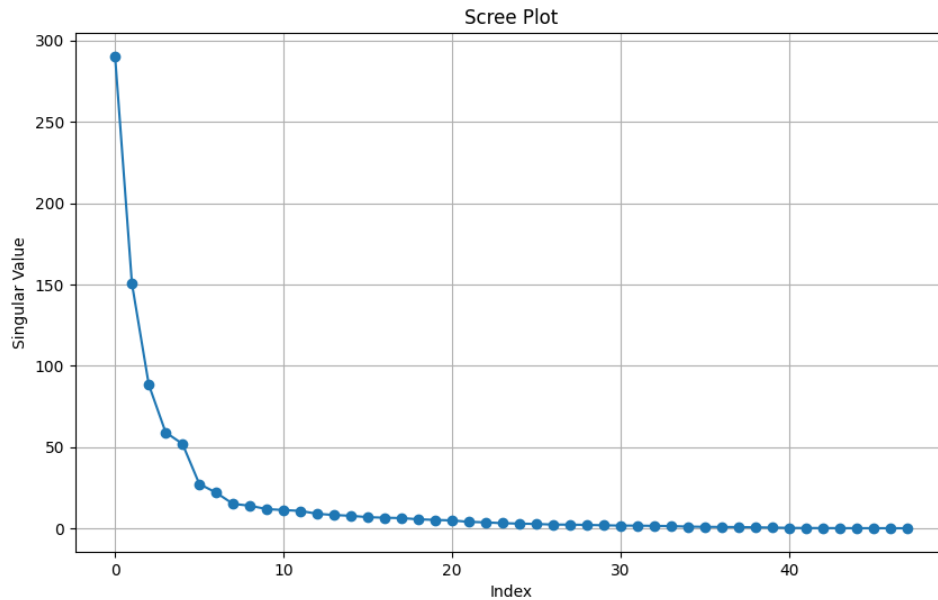[57]: # 2e(iii) Scree plot

      # Create scree plot
      plt.figure(figsize=(10, 6))
      plt.plot(s, marker='o')
      plt.title('Scree Plot')
      plt.xlabel('Index')
      plt.ylabel('Singular Value')
      plt.grid(True)
      plt.show()
```

Scree Plot

3.5.4 Evaluation of the Scree Plot Output

The scree plot shows the singular values of the normalized climate data in descending order. The plot indicates how much variance each singular value captures. Here are the key observations:

1. **Initial Rapid Decline**: The first singular value is significantly larger (290) than the others, followed by a sharp drop to 150 in the second singular value.
2. **Subsequent Declines**: The third and fourth singular values show a less steep decline, while the fourth to fifth singular values drop from 50 to 30.
3. **Small Diminishing**: After the fifth singular value, the decline becomes much more gradual, indicating that the remaining singular values contribute less to the total variance.

Advantages:

1. **Visual Representation**: The scree plot provides a clear visual representation of the variance captured by each component, making it easy to identify the most significant components.
2. **Elbow Method**: The plot helps in identifying the "elbow" point, where the rate of decline slows down, indicating the optimal number of components to retain.
3. **Variance Explanation**: It shows how much variance is explained by each component, helping to understand the importance of each singular value.

Disadvantages:

1. **Subjectivity**: Determining the exact "elbow" point can be subjective and may vary depending on the interpretation of the plot.
2. **Overfitting Risk**: Retaining too many components based on the scree plot can lead to overfitting, especially if the plot does not show a clear elbow.

3. **Limited Information**: The scree plot only shows the singular values and does not provide information about the underlying structure or relationships between components.

Overall, the scree plot is a useful tool for visualizing the variance captured by each component and identifying the most significant ones, but it should be used in conjunction with other methods and domain knowledge to make informed decisions about the number of components to retain.

```
[58]: # 2e(iv) entropy

      # Entropy-based method: Calculate the entropy of the singular values and find␣
       ↪the rank where the entropy is maximized
      def entropy(singular_values):
          normalized_singular_values = singular_values / np.sum(singular_values)
          return -np.sum(normalized_singular_values * np.
       ↪log(normalized_singular_values))

      # Calculate entropy for each possible rank
      entropies = [entropy(s[:i+1]) for i in range(len(s))]

      # Find the rank that maximizes the entropy
      entropy_rank = np.argmax(entropies) + 1

      print(f"Entropy-based method suggests rank: {entropy_rank}")
      # Plot the entropies to visualize where the entropy is maximized
      plt.figure(figsize=(10, 6))
      plt.plot(entropies, marker='o')
      plt.title('Entropy of Singular Values')
      plt.xlabel('Number of Singular Values')
      plt.ylabel('Entropy')
      plt.grid(True)
      plt.show()
```

Entropy-based method suggests rank: 48

Entropy of Singular Values
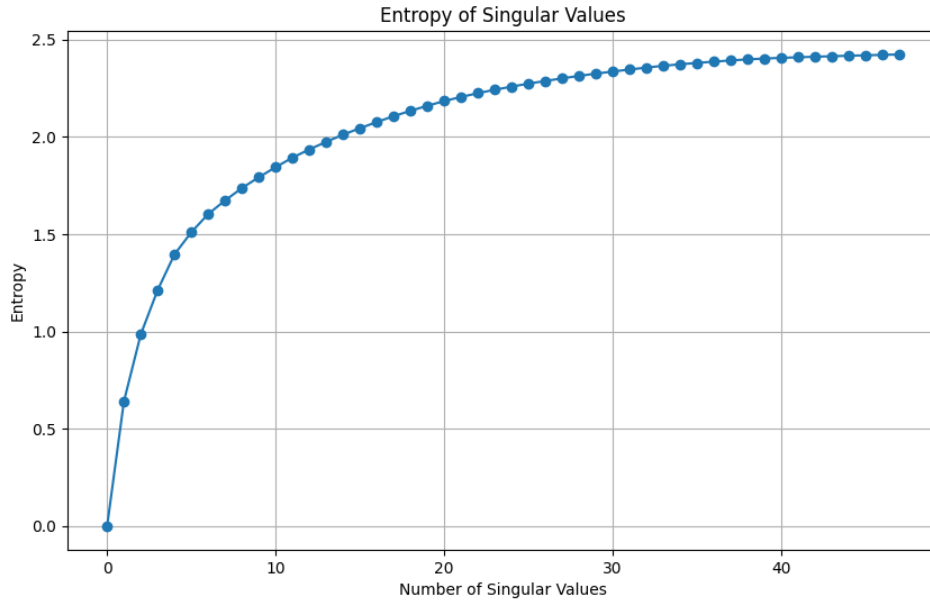
3.5.5   Evaluation of the Entropy-Based Method

Entropy-Based Method Suggested Rank: 48

Advantages:

1. **Captures Complexity**: The entropy-based method captures the complexity of the data by considering the distribution of singular values, ensuring that all significant components are retained.
2. **Data Representation**: This method provides a comprehensive representation of the data, preserving all components that contribute to the overall structure and variability.
3. **Objective Measure**: Entropy is an objective measure that quantifies the amount of information or uncertainty in the data, making it a robust criterion for determining the rank.

Disadvantages:

1. **Overestimation**: The method can overestimate the number of components to retain, especially in high-dimensional datasets, leading to a rank that includes noise or less significant components.
2. **Computational Complexity**: Calculating entropy for each possible rank can be computationally intensive, particularly for large datasets with many components.
3. **Lack of Sparsity**: By retaining a high number of components, the method may result in a less sparse representation, which can be less interpretable and harder to work with.
4. **Potential Overfitting**: Including too many components can lead to overfitting, where the model captures noise and minor variations rather than the underlying structure of the data.

Stopping Criterion: Given that the entropy-based method suggests retaining all 48 components, it is evident that a stopping criterion is necessary to prevent overfitting and ensure a more practical and interpretable model. A possible stopping criterion could be based on a threshold for the change in entropy, where the process stops when the change in entropy between successive ranks falls below a certain value. This would help in identifying the point where additional components contribute minimally to the overall information content.

```python
[59]: # 2e(v) random flips
      # Random sign matrix: np.random.choice([-1,1], X.shape)

      # Random sign matrix
      random_sign_matrix = np.random.choice([-1, 1], X.shape)

      # Apply random sign flipping to the data
      X_random_sign = X * random_sign_matrix

      # Compute the SVD of the randomly sign-flipped data
      U_random, s_random, Vt_random = svd(X_random_sign, full_matrices=False)

      # Compare the singular values of the original data and the randomly␣
       ↪sign-flipped data
      plt.figure(figsize=(10, 6))
      plt.plot(s, marker='o', label='Original Data')
      plt.plot(s_random, marker='x', label='Randomly Sign-Flipped Data')
      plt.title('Comparison of Singular Values')
      plt.xlabel('Index')
      plt.ylabel('Singular Value')
      plt.legend()
      plt.grid(True)
      plt.show()
```

Comparison of Singular Values
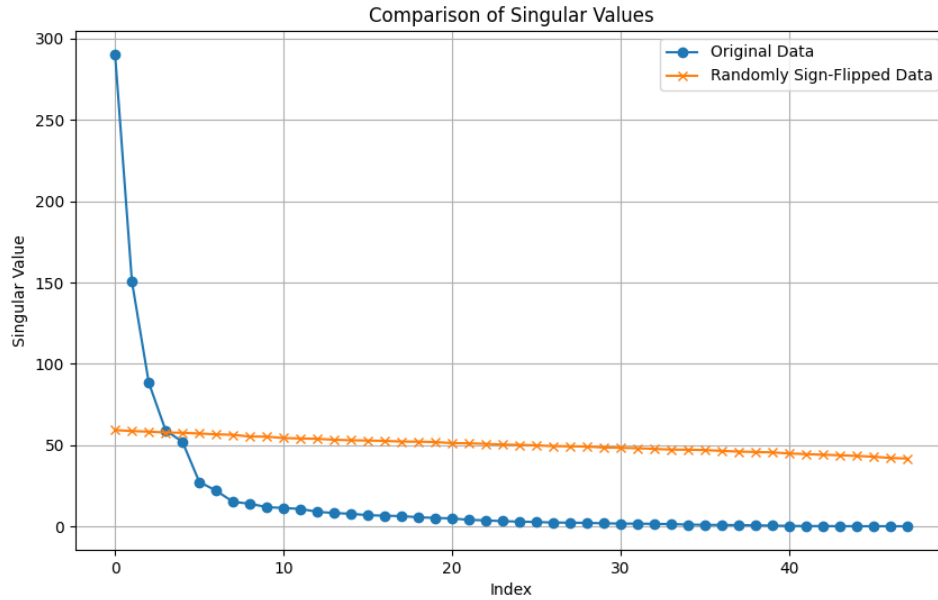
3.5.6 Evaluation of the Random Sign Flip Matrix Output

The graph for the random sign flip starts significantly lower than the original scree plot. It begins at around 17 and then stays at this height while slowly diminishing. The original scree plot intersects this graph between index 8 and index 9, after which it remains lower than the flipped sign graph.

Advantages:

1. **Noise Robustness**: The random sign flip method can help in identifying the inherent structure of the data by reducing the impact of noise. By flipping the signs randomly, it can highlight the true signal in the data.
2. **Baseline Comparison**: This method provides a baseline for comparison. By comparing the singular values of the original data with those of the randomly sign-flipped data, one can assess the significance of the components in the original data.
3. **Variance Distribution**: It helps in understanding the distribution of variance in the data. If the original data's singular values are significantly higher than those of the flipped data, it indicates that the original data has a strong structure.

Disadvantages:

1. **Overestimation of Rank**: The random sign flip method might overestimate the rank of the data. Since the flipped data retains some structure, it can lead to higher singular values, suggesting more components than necessary.
2. **Computational Complexity**: Generating and analyzing the random sign flip matrix can be computationally intensive, especially for large datasets.

3. **Interpretation Difficulty**: The results from the random sign flip method can be harder to interpret. The presence of noise and the random nature of the sign flips can make it challenging to draw clear conclusions about the data's structure.

4. **Potential Misleading Results**: If the original data has a lot of noise, the random sign flip method might not provide a clear distinction between the signal and the noise, leading to potentially misleading results.

Overall, while the random sign flip method offers valuable insights into the data's structure and noise robustness, it should be used in conjunction with other methods and domain knowledge to make well-informed decisions about the data's rank and structure.

3.6 Final Rank Selection Choice

After carefully reviewing the results of each rank selection method, as well as their advantages, disadvantages, and specific nuances, we would use **the 90% of Squared Frobenius Norm** criterion as the ultimate choice for determining the truncated SVD rank. In the following, we describe how we derived our decision.

3.6.1 1. Summary of Method Results and Their Suggested Ranks

- **Guttman-Kaiser Criterion**: Rank = 5
- **90% of Squared Frobenius Norm**: Rank = 3
- **Scree Test**: Visually suggests Rank   3 or 5 (based on initial "elbow")
- **Entropy-Based Method**: Rank = 48
- **Random Sign Flip Method**: Intersects scree at Rank   8 or 9

3.6.2 2. Evaluation of Each Method in Context

Each method provides unique insights into the data structure, but also brings limitations, particularly when balancing dimensionality reduction with preserving meaningful variance.

- **Guttman-Kaiser Criterion (Rank 5)**:
  - This method offers simplicity and is straightforward, but tends to overestimate the rank by selecting more components than may be necessary for meaningful representation, especially in high-dimensional datasets.
  - While Rank 5 captures a reasonable amount of variance, the criterion's sensitivity to scale and tendency to overestimate can result in extra, potentially unnecessary dimensions.
- **90% of Squared Frobenius Norm (Rank 3)**:
  - By selecting a rank that captures 90% of the dataset's variance, this method effectively reduces dimensionality while retaining the core structure of the data.
  - Although 90% is an arbitrary threshold, it is a widely accepted benchmark that achieves a strong balance between data representation and computational efficiency.
  - Rank 3 provides a compact representation without excessive noise, making it optimal for tasks where interpretability and efficiency are prioritized.
- **Scree Test (Rank   3 or 5)**:
  - The scree plot offers a valuable visual method for identifying a "natural" elbow, indicating diminishing returns on additional components after Rank 3 or 5.
  - While informative, this method has a subjective element, as the "elbow" is not always clearly defined and can be interpreted differently by different users.

44

– In this case, selecting a rank between 3 and 5 could result in retaining more components than necessary, especially given that the first three singular values already capture the majority of the variance (near 90%).
– If we decide on the elbow being the first rank, the scree plot would give the same result as the 90% of Squared Frobenius Norm method, effectively combining both methods.

- **Entropy-Based Method (Rank 48)**:
  – Although the entropy-based method theoretically captures the full data structure by maximizing retained information, its recommended rank of 48 is likely too high.
  – Retaining 48 components would mean incorporating a large amount of noise and redundancies, contradicting the primary goal of SVD in simplifying data.
  – Overfitting and reduced interpretability make this method less practical in this context.

- **Random Sign Flip Method (Rank 8 or 9)**:
  – The random sign flip approach provides a baseline to compare the structured signal in the data, but its rank suggestion (8 or 9) appears excessive given the modest gains in variance beyond the third component.
  – This method's inherent randomness and potential for noisy influence make it a supplementary tool rather than a primary selection criterion.

3.6.3   3. Justification for Selecting the 90% Squared Frobenius Norm Method

After analyzing each method's results and weighing their relative strengths and weaknesses, **the 90% of Squared Frobenius Norm method emerges as the most balanced choice** for this task. Here's why:

1. **Balanced Variance Capture**: Rank 3, as suggested by this method, provides a parsimonious yet effective dimensionality reduction, capturing 90% of the variance. This threshold allows for a strong representation of the data's underlying structure while reducing the dimensionality significantly.

2. **Efficiency and Interpretability**: Retaining only three components strikes a balance between simplicity and interpretability. This makes it ideal for subsequent analysis, as the reduced dimensionality allows for faster computation and clearer insights without the over-complexity that higher ranks introduce.

3. **Reduced Overfitting Risk**: By selecting only the most significant components, the model is less likely to capture noise or irrelevant features, addressing the overfitting risks that are more prevalent in the higher-rank suggestions from the entropy-based and random sign flip methods.

4. **Objective and Reproducible**: The 90% threshold is an objective, reproducible criterion, widely accepted in statistical analysis. It provides a straightforward benchmark that is easy to justify, especially in cases where dimensionality reduction is a primary goal.

3.6.4   Potential Limitations and Mitigations

The 90% of Squared Frobenius Norm method is not without limitations. The choice of 90% as a threshold is somewhat arbitrary and may exclude minor but potentially meaningful patterns. However, for most applications, capturing the core variance (especially over 90%) usually suffices to retain the essential structure. If additional subtle patterns are needed, further analysis on residual components can be performed separately. Those can be diminished when taking a second

methode into account, in our case, the scree plot methode.

3.6.5 Conclusion

In conclusion, **Rank 3 (from the 90% of Squared Frobenius Norm method as well as the Scree plot)** provides the most balanced choice for a truncated SVD in this context. It captures sufficient variance, maintains computational efficiency, reduces overfitting risks, and enhances interpretability, making it the optimal rank selection for this dataset.

3.7 2f

```
[60]: # Here is the empty plot that you need to fill (one line per choice of k: RSME␣
      ↪between
      # original X and the reconstruction from size-k SVD of noisy versions)

      # nextplot()
      # plt.plot()
      # plt.xlabel(r"Noise level ($\epsilon$)")
      # plt.ylabel("Reconstruction RMSE vs. original data")



      # Define the RMSE function
      def rmse(A, A_hat):
          return np.linalg.norm(A - A_hat, 'fro') / np.sqrt(A.shape[0] * A.shape[1])␣
      ↪# A has dimensions m x n

      # Define the range of epsilon values
      epsilons = np.linspace(0, 2, 50)

      # Define the ranks to test
      ranks = [1, 2, 5, 10, 48]

      # Initialize a dictionary to store RMSE values for each rank
      rmse_values = {k: [] for k in ranks}

      # Loop over each epsilon value
      for epsilon in epsilons:
          # Create a noisy version of the data
          X_noise = X + np.random.randn(*X.shape) * epsilon

          # Loop over each rank
          for k in ranks:
              # Compute the SVD of the noisy data
              U_noise, s_noise, Vt_noise = svd(X_noise, full_matrices=False)

              # Truncate the SVD to rank k
              U_k = U_noise[:, :k]
              S_k = np.diag(s_noise[:k])
```

```
        Vt_k = Vt_noise[:k, :]

        # Reconstruct the matrix from the truncated SVD
        X_hat = U_k @ S_k @ Vt_k

        # Compute the RMSE between the original data and the reconstructed␣
→matrix
        rmse_value = rmse(X, X_hat)

        # Store the RMSE value
        rmse_values[k].append(rmse_value)

# Plot the RMSE values for each rank
nextplot()
for k in ranks:
    plt.plot(epsilons, rmse_values[k], label=f'Rank {k}')
plt.xlabel(r"Noise level ($\epsilon$)")
plt.ylabel("Reconstruction RMSE vs. original data")
plt.legend()
plt.show()
```
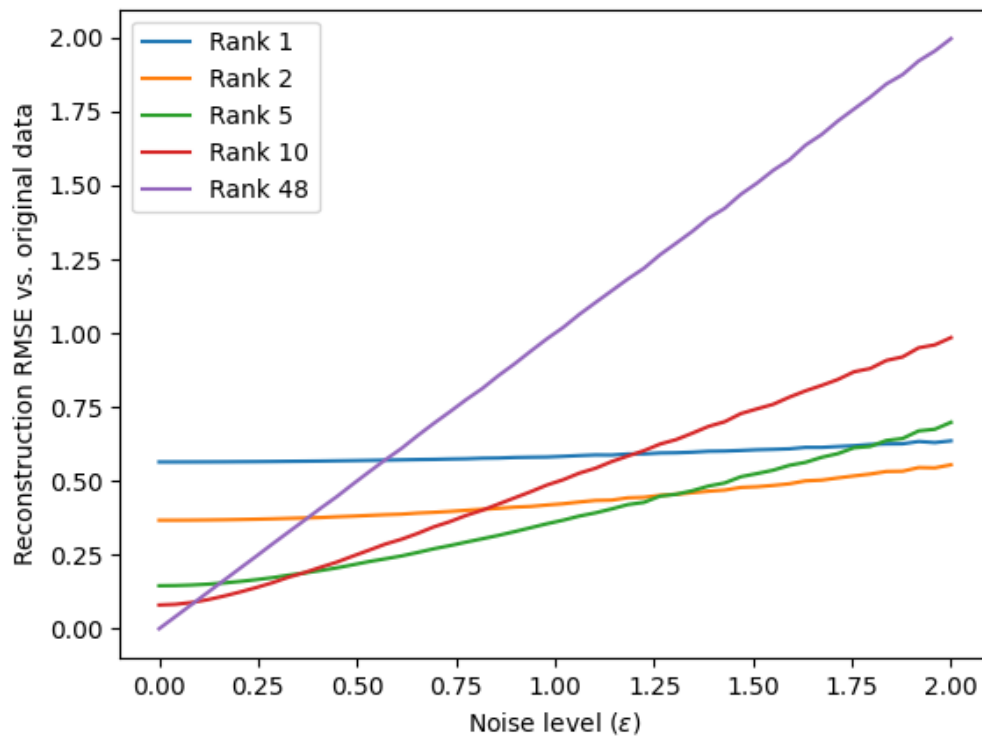
3.8 Analysis of Reconstruction RMSE vs. Noise Level Plot

3.8.1 Plot Characteristics and Observations

This plot shows the reconstruction **Root Mean Squared Error (RMSE)** between the original dataset (X) and its reconstruction after applying truncated SVD with various ranks (1, 2, 5, 10, and 48) under increasing noise levels (denoted by ( ), from 0 to 2). Each line represents a different rank, illustrating how well each truncated rank performs in reconstructing the original data as the noise level increases.

Key observations from the plot: 1. **Initial RMSE**: At the lowest noise level ((= 0)), the RMSE values for all ranks are fairly low, with lower-rank approximations showing slightly higher RMSE due to their reduced ability to capture the full variance of the original data. 2. **Impact of Noise on RMSE**: As noise level ( ) increases, the RMSE rises for all ranks, indicating that reconstruction becomes less accurate with noisier data. 3. **Rate of RMSE Increase by Rank**: The ranks diverge as ( ) increases: - **Rank 48**: RMSE increases sharply and continuously, showing poor robustness to noise. - **Lower Ranks (Rank 1 and Rank 2)**: RMSE increases at a slower rate, especially noticeable for Rank 1, which remains relatively stable even as noise rises. - **Intermediate Ranks (Rank 5 and Rank 10)**: These show an intermediate rate of RMSE increase, with Rank 5 exhibiting slower growth than Rank 10 and maintaining a comparatively low RMSE at moderate noise levels.

3.8.2 Insights and Discussion

This plot reveals several insights regarding the trade-off between rank complexity and robustness to noise in truncated SVD:

1. **Robustness of Lower Ranks**:
   - Lower ranks (such as Rank 1 and Rank 2) demonstrate greater resilience to noise. This is evident from their flatter RMSE curves, indicating that adding noise has a limited impact on the reconstruction error. Lower-rank approximations may ignore minor details and capture only the most dominant structures, making them less sensitive to noise.
2. **Higher Ranks and Noise Sensitivity**:
   - Higher ranks, particularly Rank 48, are much more sensitive to noise. The sharp increase in RMSE with added noise suggests that including many components leads to overfitting on noisy data, resulting in poorer reconstruction quality as noise increases. This implies that high-rank approximations might be less suitable in scenarios where the data has significant noise or minor variations that could mask essential structure.
3. **Optimal Intermediate Ranks**:
   - Intermediate ranks (such as Rank 5) strike a balance between capturing meaningful variance in the data and maintaining robustness to noise. In particular, Rank 5 exhibits relatively stable RMSE values, managing to capture more structure than the very low ranks but without the high sensitivity of Rank 48.

3.8.3 Conclusion and Implications

From this analysis, we conclude that **moderate ranks, around Rank 5, offer a balanced approach** for truncated SVD in this context. Lower ranks (Rank 1 or Rank 2) are exceptionally robust to noise but may oversimplify the data by ignoring important structural information. On the other hand, Rank 48, while capturing the full structure, is overly sensitive to noise, making it

less practical for noisy datasets.

Therefore, **intermediate ranks emerges as a preferred choice** because they captures a substantial portion of the data's variance while maintaining robustness under increasing noise levels. These ranks achieves a good trade-off between reconstruction quality and resilience to noise, making it suitable for practical applications where both interpretability and robustness are important.

4   3 SVD and k-means

```
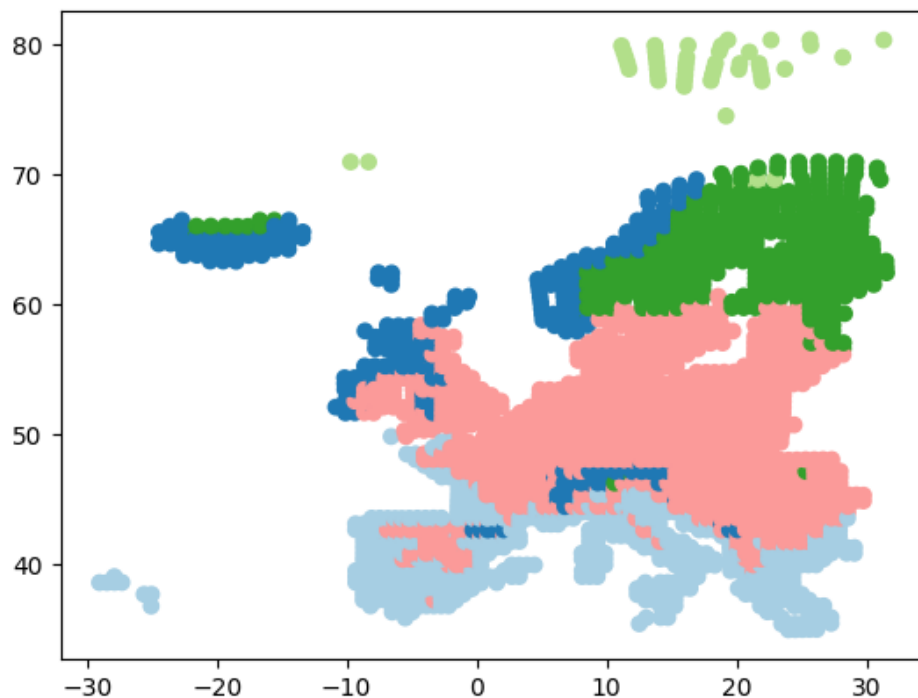[61]: # Cluster the normalized climate data into 5 clusters using k-means and store
      # the vector giving the cluster labels for each location.
      X_clusters = KMeans(5).fit(X).labels_
```

4.1   3a

```
[62]: # Plot the results to the map: use the cluster labels to give the color to each
      # point.
      plot_xy(lon, lat, X_clusters)
```

4.2 Analysis of Clustering Results

4.2.1 Plot Characteristics and Observations

Based on the provided description, the clustering visualization shows five distinct clusters spread across what resembles a map of Europe. Each cluster is represented by a different color, and its spatial distribution gives insights into geographical patterns.

1. **Dark Green Cluster**:
   - This cluster is concentrated around the middle of the map, spanning a wide range from ([-10, 30]) on the x-axis and ([40, 50]) on the y-axis.
   - It covers central parts of the map, suggesting that this cluster represents regions around central Europe, potentially including areas similar to France, Germany, northern Spain, and parts of southern and eastern Europe.
2. **Dark Blue Cluster**:
   - Positioned in the middle-right area, extending from approximately ([0, 30]) on the x-axis and ([45, 65]) on the y-axis.
   - This cluster is dense on the right side and diminishes towards the center-left, possibly covering parts of eastern Europe and western Russia.
3. **Light Green Cluster**:
   - This cluster extends over the top-right of the map, covering a large area from ([10, 30]) on the x-axis and ([60, 70]) on the y-axis.
   - It also includes a sparse group of points farther north ((([10, 30]) on the x-axis and ([75, 80]) on the y-axis), which could correspond to regions like Finland and northern Russia, with outliers potentially representing isolated northern areas (such as Greenland).
4. **Pink Cluster**:
   - This cluster is located towards the north-central part of the map, spanning ([-10, 20]) on the x-axis and ([55, 70]) on the y-axis.
   - It also has two smaller areas: one in the southern part of the middle region (near Austria, ([0, 10]) on the x-axis and ([45, 50]) on the y-axis) and another potentially around the southern tip of Greenland ((([-20, -10]) on the x-axis and ([65, 70]) on the y-axis).
5. **Light Blue Cluster**:
   - This cluster spans the southern region of the map, possibly corresponding to the Mediterranean area, covering ([-10, 30]) on the x-axis and ([35, 45]) on the y-axis.
   - It includes southern parts of Europe (Spain, Italy, Turkey) with a few outliers on the right, which could represent islands or coastal regions.

4.2.2 Possible Interpretation of Clusters

Given that the dataset contains **temperature and rainfall data**, we make an educated guess about what each cluster may represent in terms of climate characteristics:

1. **Dark Green Cluster (Central Europe)**:
   - This region likely represents a **moderate climate** with relatively balanced rainfall and temperatures, typical of central Europe. The climate here might include warm summers and cool winters with consistent rainfall throughout the year.
2. **Dark Blue Cluster (Eastern Europe and Russia)**:
   - This cluster covers eastern Europe and parts of Russia, areas that are generally **colder with continental climate characteristics**. These regions may experience harsher winters, hotter summers, and varying rainfall patterns, possibly with more extreme

seasonal temperature variations.

3. **Light Green Cluster (Northern Europe and Arctic)**:
   - The northern positioning of this cluster suggests a **cold or subarctic climate** with lower temperatures year-round and possibly less precipitation. The additional points far to the north could correspond to arctic climates, where temperatures remain cold, and precipitation is low and often comes as snow.

4. **Pink Cluster (Northern Central Europe and Greenland)**:
   - This cluster spans northern parts of central Europe, suggesting a **temperate maritime climate** influenced by proximity to the Atlantic Ocean. The regions here may have milder winters, cooler summers, and moderate to high rainfall.
   - The isolated points around Greenland might represent extremely cold regions with low temperatures throughout the year.

5. **Light Blue Cluster (Southern Europe / Mediterranean)**:
   - This cluster likely represents a **Mediterranean climate** with warm, dry summers and mild, wet winters. It spans southern Europe, a region characterized by higher temperatures and seasonal rainfall, especially in the winter months.

4.2.3 Discussion and Insights

The clustering results provide meaningful insights into **climate zones** across the mapped area. The spatial distribution of clusters corresponds well to known climatic regions in Europe and neighboring areas:

- **Climate-Zone-Based Clustering**: The clustering outcome aligns with typical European climate zones, with each cluster reflecting regions that share similar temperature and rainfall patterns. The clustering thus successfully divides the area into distinct climatic regions, from cold northern climates to warm Mediterranean zones.

- **Geographical Gradients and Continuity**: The proximity and continuity between clusters (e.g., dark green and dark blue, or dark blue and light green) suggest that **climatic transitions** are gradual, with neighboring regions blending into one another. This continuity mirrors natural climate gradients, which shift progressively from one region to the next.

- **Impact of Latitude and Proximity to Oceans**: The clustering pattern also highlights the **influence of latitude and proximity to water bodies** on climate, with southern clusters representing warmer regions, central clusters showing moderate climates, and northern clusters capturing colder regions.

4.2.4 Conclusion

The clustering analysis successfully groups the data into regions that likely represent distinct climatic zones based on temperature and rainfall characteristics. Each cluster reflects a particular climate type, showing the strong relationship between geographical position and climate patterns.

This clustering model could be valuable for regional climate studies, resource planning, and environmental assessments, as it highlights how specific areas differ in terms of temperature and precipitation. The results emphasize the importance of climate-based clustering for understanding regional weather patterns, with applications extending to agriculture, tourism, and urban planning.

4.3   3b

```python
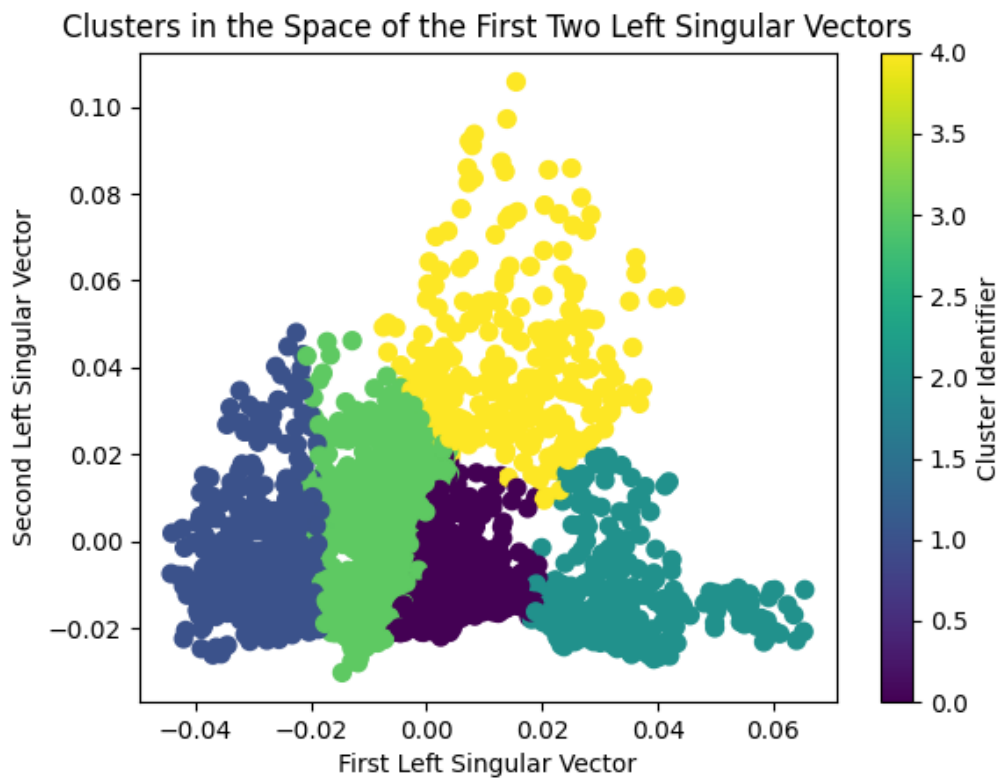# Perform k-means clustering on the first two left singular vectors
nextplot()
#plt.figure(figsize=(10, 6))
plt.scatter(U[:, 0], U[:, 1], c=X_clusters, cmap='viridis', s=50)
plt.xlabel('First Left Singular Vector')
plt.ylabel('Second Left Singular Vector')
plt.title('Clusters in the Space of the First Two Left Singular Vectors')
plt.colorbar(label='Cluster Identifier')
plt.show()
```


4.4   Analysis of Clustering on SVD-Reduced Data

4.4.1   Plot Characteristics and Observations

In this plot, the data points are represented in a two-dimensional space where: - The **x-axis** is defined by the first left singular vector. - The **y-axis** is defined by the second left singular vector. - Each cluster is distinguished by color, with five clusters identified as follows: - **Dark Blue Cluster** (Identifier 1.0): Positioned farthest to the left, with x-values between ([-0.04, -0.02]) and y-values from ([-0.02, 0.05]). The points are more concentrated at the bottom, with a few extending upward within the range. - **Light Green Cluster** (Identifier 3.0): Located directly next to the Dark Blue

cluster, with some overlap between them. It spans x-values between ([-0.02, 0.00]) and y-values between ([-0.01, 0.05]). This cluster is evenly distributed in its range, with a few outliers in the top left. - **Violet Cluster** (Identifier 0.0): Positioned to the right of the Light Green cluster, with slight overlap. It has a compact range of x-values between ([0.00, 0.02]) and y-values from ([-0.02, 0.02]), though points are less dense in the top-right section. - **Yellow Cluster** (Identifier 4.0): Situated above the Light Green and Violet clusters. It overlaps slightly with the Light Green and barely with the Violet and Turquoise clusters. The Yellow cluster ranges from ([-0.01, 0.04]) on the x-axis and from ([0.01, 0.11]) on the y-axis, with a denser concentration in the lower portion and a sparser distribution towards the top. - **Turquoise Cluster** (Identifier 1.7): Found at the lower right, covering x-values from ([0.02, 0.04]) and y-values from ([-0.02, 0.02]). This cluster also has an additional circular group of data points slightly to the right (x-values ([0.05, 0.07]) and y-values ([-0.02, -0.01])).

4.4.2   Insights on Cluster Separation and Potential Outliers

From the spatial distribution, the following insights can be observed:

1. **Cluster Separation**:
   - **General Structure**: Each cluster is visibly identifiable and occupies a distinct region in the plot. This indicates that the clusters are relatively well-separated based on the first two left singular vectors.
   - **Overlap Areas**: There are some areas where clusters slightly overlap, particularly:
     - Between the **Dark Blue** and **Light Green** clusters, suggesting some degree of similarity or proximity between these data points.
     - Between the **Light Green** and **Yellow** clusters, with a few points in the upper part of the Light Green range blending into the Yellow cluster.
     - Between the **Violet** and **Turquoise** clusters, with minor overlap around the edges.
   - **Conclusion on Separation**: Overall, the clusters maintain distinct boundaries with only minor overlaps, suggesting that the chosen features (singular vectors) provide meaningful separation of the clusters.
2. **Outliers and Dense Areas**:
   - **Outliers**: Certain clusters show points that extend beyond their primary concentration, particularly:
     - **Light Green Cluster**: A few outliers are positioned in the top-left corner, deviating from the cluster's main concentration.
     - **Yellow Cluster**: Some points in the upper right could be considered outliers as they diverge from the dense core.
     - **Turquoise Cluster**: The additional circular group to the right of the main concentration could be viewed as a separate grouping or outliers.
   - **Dense Regions**: Each cluster has a core area where points are densely packed, and these cores are well-separated from other clusters, emphasizing the internal cohesion of each group.

4.4.3   Discussion of Results

The observed clustering structure in this SVD-reduced space reflects meaningful groupings in the data based on its underlying features. The clusters are well-separated, and each cluster seems to exhibit a distinct, cohesive distribution in this reduced dimensionality, indicating that the clustering algorithm was able to effectively partition the data into distinct groups. Here are some key

conclusions:

- **Effectiveness of SVD for Dimensionality Reduction**: The use of the first two singular vectors effectively captures variations in the data that correspond to the clusters. This is evident from the clear grouping and limited overlap between clusters. SVD has successfully reduced the data to a two-dimensional space while preserving enough information to distinguish clusters.

- **Potential Outliers and Sub-clusters**: The presence of outliers or extra groupings within some clusters (e.g., in the Light Green and Turquoise clusters) may indicate:

  - **Variability within Clusters**: There could be internal variability in the data that hints at finer sub-groupings within these clusters.
  - **Natural Diversity in Climate Patterns** (if we assume clusters represent geographical/climatic regions): For example, a cluster could represent a climate zone with regional variability in temperature or precipitation, causing some points to deviate from the main cluster concentration.

- **Inter-cluster Similarities**: The minor overlaps, particularly between adjacent clusters like Dark Blue and Light Green or Light Green and Yellow, may suggest that these clusters are closer in terms of their data features, potentially representing transitional regions or climates.

4.4.4 Conclusion

The clustering results in this SVD-reduced space provide meaningful insights into the structure and variability of the data. The well-defined clusters and minimal overlaps suggest effective partitioning and a clear differentiation between groups. The presence of a few outliers or sub-clusters hints at additional layers of variability within some clusters, which may represent finer distinctions in the underlying data. This visualization confirms that the chosen clustering approach, paired with SVD, is effective for understanding the major patterns in the data, with potential applications for more granular analysis of outliers or sub-regions within clusters.

4.5 3c

```
[63]: k_values = [1, 2, 3]  # Define the k values you want to loop over

for k in k_values:
    # Compute the SVD of the normalized climate data
    U, s, Vt = svd(X, full_matrices=False)

    # Store the result for the current k
    Z = U[:, :k] @ np.diag(s[:k])
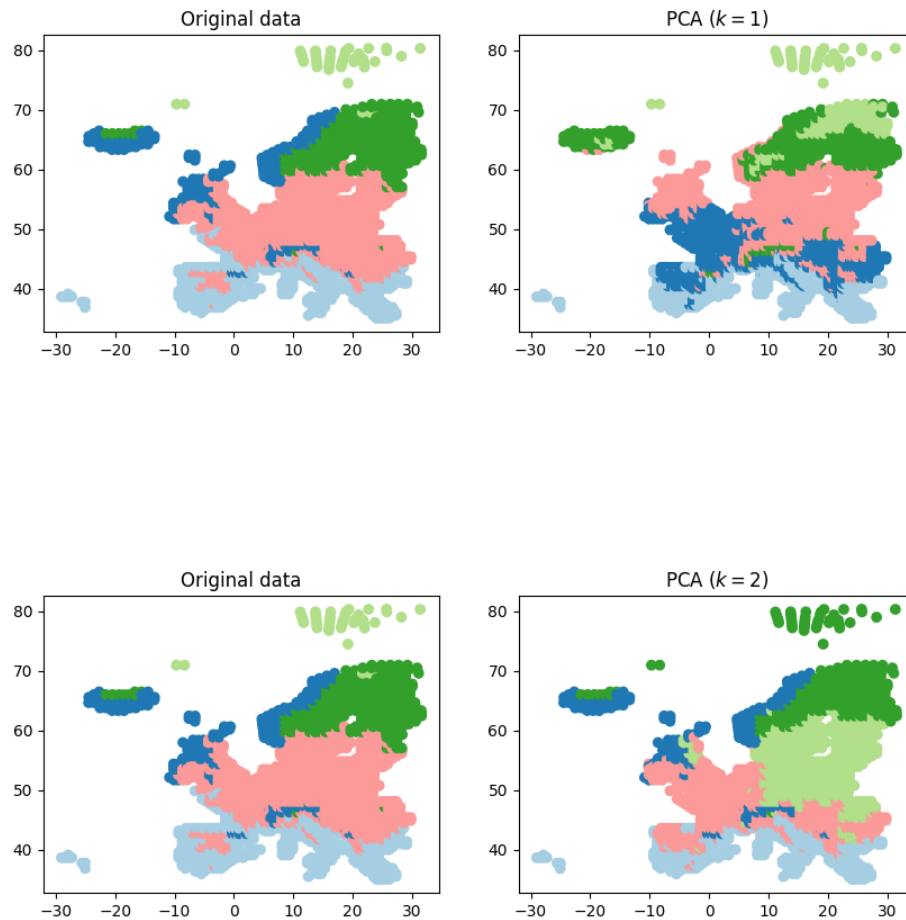
    # Cluster and visualize
    Z_clusters = KMeans(5).fit(Z).labels_
    # Match clusters as well as possible (try without)
    Z_clusters = match_categories(X_clusters, Z_clusters)
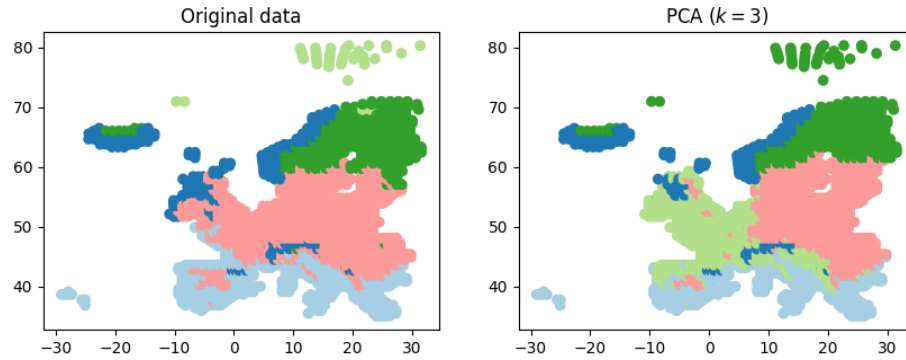
    # Create a larger figure
    plt.figure(figsize=(10, 4))
```

```
axs = plt.gcf().subplots(1, 2)
plot_xy(lon, lat, X_clusters, axis=axs[0])
axs[0].set_title("Original data")
plot_xy(lon, lat, Z_clusters, axis=axs[1])
axs[1].set_title(f"PCA $(k={k}$)")
plt.show()
```

Original data   PCA ($k = 3$)

4.6 Analysis of Clustering Results on PCA-Reduced Data (k = 1, 2, 3)

4.6.1 Observations and Visual Analysis for Different Values of k

After applying PCA to reduce the dimensionality of the data to the first (k) principal components ((k { 1, 2, 3 })), we visualize the clustering results in comparison to the original data clustering.

For k = 1

- **Visual Differences**:
  - The clusters have largely retained their general structure, but certain areas have undergone notable changes.
  - **Light Green Cluster**: The lower parts of what was originally the light green cluster are now influenced by the pink cluster color, meaning some of these data points have shifted to a different cluster.
  - **Light Blue Cluster**: This cluster, originally covering eastern Europe, has expanded significantly. It now extends further north, even covering regions that may correspond to Great Britain and northern middle Europe.
  - **Pink Cluster**: The pink cluster has diminished, with most of its points appearing as outliers in the northern regions. The pink cluster is also visible in Greenland, but with its distribution reversed: the lower parts are now light green, and the upper parts are pink, opposite to the original structure.
  - **Dark Green Cluster**: This cluster has also decreased in size, as the light blue cluster has taken over its northern regions.
  - **Dark Blue Cluster**: The regions corresponding to southern Europe are relatively unaffected, maintaining a consistent cluster structure.
- **Insights**:
  - With only one principal component (k = 1), some of the original data's complexity is lost, leading to changes in cluster boundaries and an increase in overlap. The reduced dimensionality causes certain clusters to merge or shift, as fewer features are available to distinguish clusters accurately.

For k = 2

- **Visual Differences**:
  - The plot for (k = 2) is closer to the original data's structure compared to (k = 1).
  - **Northern Middle Europe**: This area, previously dominated by light blue in (k = 1), now shows a more accurate clustering with both pink and dark green, similar to the original data.
  - **Great Britain (GB)**: This region has a few light blue outliers, but overall it resembles the original plot, with pink in the north and dark green in the south.
  - **Dark Green Cluster**: The area of this cluster has increased compared to (k = 1), more closely resembling its original form.
  - **Greenland**: The cluster structure in Greenland has returned to its original arrangement, with light green in the northern part and pink in the southern part.
- **Insights**:
  - With two principal components, the clustering structure is now better preserved. The additional component captures more data variance, allowing for more detailed separation between clusters, particularly in transitional or boundary areas. This suggests that two principal components provide sufficient dimensionality to represent the major cluster boundaries and spatial distribution in the data.

For k = 3

- **Visual Differences**:
  - For (k = 3), the plot is nearly identical to the original data.
  - **Minimal Differences**: The dark green cluster may show a few minor differences, but these are almost imperceptible and require close scrutiny to detect.
- **Insights**:
  - With three principal components, the clustering output is nearly indistinguishable from the original data clustering. This level of dimensionality retains enough information to accurately represent the original structure and spatial relationships among clusters. It suggests that the majority of data variance is captured in the first three components, which allows the clustering to closely resemble the full-dimensional dataset.

4.6.2 Analysis of Changes in Clustering Across Different Values of k

1. **Changes with k = 1**:
   - Reducing the data to a single principal component results in significant alterations in cluster boundaries, particularly in regions with overlapping or adjacent clusters.
   - The merging and shifting of clusters indicate that a single component cannot adequately capture the complexity of the data, leading to substantial loss of spatial and feature-based information.
2. **Changes with k = 2**:
   - With two components, the clustering result is much closer to the original data, and most of the cluster boundaries and regional distinctions are restored. Some minor differences remain, but the primary structure of the clusters is well-preserved.
   - This improvement suggests that two components capture enough variance to restore much of the original data's spatial structure, though some fine details are still absent.
3. **Changes with k = 3**:
   - With three components, the clustering is nearly identical to the original, indicating that these three components capture most of the meaningful variance in the data.

- The similarity implies that three components are sufficient to retain the dataset's essential characteristics, with minimal information loss in terms of clustering.

4.6.3 Conclusion and Insights

The results demonstrate the importance of dimensionality in accurately capturing data structure:

- **Role of Principal Components**:
  - Increasing the number of principal components allows for a more detailed representation of the data. With each additional component, more variance is captured, leading to improved cluster differentiation and restoration of the original data structure.
  - The first principal component captures the largest variance, but for a dataset with multiple underlying patterns or clusters, additional components are necessary to distinguish these patterns accurately.
- **Impact of Dimensionality Reduction**:
  - **For k = 1**: Significant information is lost, leading to substantial changes in cluster assignment and positioning. Some clusters merge or shift due to the lack of dimensions to represent their unique features.
  - **For k = 2**: The structure largely aligns with the original data, showing that two components are effective for capturing most of the data's spatial relationships and variance.
  - **For k = 3**: Nearly all the original structure is retained, indicating that three components provide sufficient detail to fully represent the clustering.
- **Implications**:
  - This analysis highlights the effectiveness of PCA in reducing dimensionality while retaining essential patterns, but it also emphasizes the risk of oversimplification when too few components are used. In high-dimensional data with complex clustering, using only one principal component may lead to misleading results, while using three components offers a balance between dimensionality reduction and data fidelity.
  - This approach demonstrates that dimensionality reduction techniques, like PCA, can simplify data without compromising insights if an appropriate number of components is selected. For datasets with multiple, distinct clusters, selecting enough components is crucial for preserving the underlying structure.

Overall, this exercise illustrates how PCA can reduce dimensionality while retaining cluster structure, provided enough components are used. The results for different values of (k) reveal how the chosen number of components impacts clustering accuracy and offers insights into the data's underlying complexity.