In [1]:

```
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

%matplotlib notebook
```

# 1 Intuition on SVD

In [2]:

```
M1 = np.array(
    [
        [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],
    ]
)

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],
    ]
)
```

# 1a

In [26]:

```
# YOUR PART
```

# 1b

In [4]:

```
# YOUR PART
def compute_svd(X):
    U, s, Vt = np.linalg.svd(X)
    S = np.diag(s)
    return U, S, Vt

i=1
for matrix in [M1,M2,M3,M4,M5,M6]:
    U, S, Vt = compute_svd(matrix)
    print('For Matrix {}'.format(i))
    print('Matrix U is {}'.format(U))
    print('Matrix S is {}'.format(S))
    print('Matrix Vt is {}'.format(Vt))
    i=i+1
```

```
For Matrix 1
Matrix U is [[-0.57735027 -0.57735027  0.          0.         -0.577
35027]
 [-0.57735027 -0.21132487  0.          0.          0.78867513]
 [-0.57735027  0.78867513  0.          0.         -0.21132487]
 [ 0.          0.          1.          0.          0.        ]
 [ 0.          0.          0.          1.          0.        ]]
Matrix S is [[3. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]]
Matrix Vt is [[-0.57735027 -0.57735027 -0.57735027 -0.         -0.
]
 [ 0.         -0.70710678  0.70710678  0.          0.        ]
 [ 0.          0.          0.          1.          0.        ]
 [ 0.          0.          0.          0.          1.        ]
 [ 0.81649658 -0.40824829 -0.40824829  0.          0.        ]]
For Matrix 2
Matrix U is [[ 0.          0.          0.          0.          1.
]
 [-0.57735027 -0.57735027 -0.57735027  0.          0.        ]
 [-0.57735027  0.78867513 -0.21132487  0.          0.        ]
 [-0.57735027 -0.21132487  0.78867513  0.          0.        ]
 [ 0.          0.          0.          1.          0.        ]]
Matrix S is [[5.19615242 0.         0.         0.         0.
]
 [0.         0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.        ]
 [0.         0.         0.         0.         0.        ]]
Matrix Vt is [[-0.         -0.66666667 -0.33333333 -0.66666667 -0.
]
 [ 0.          0.74535599 -0.2981424  -0.59628479  0.        ]
 [ 0.          0.         -0.89442719  0.4472136   0.        ]
 [ 0.          0.          0.          0.          1.        ]
 [ 1.          0.          0.          0.          0.        ]]
For Matrix 3
Matrix U is [[ 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]]
Matrix S is [[3.46410162 0.         0.         0.        ]
 [0.         0.         0.         0.        ]
 [0.         0.         0.         0.        ]
 [0.         0.         0.         0.        ]]
Matrix Vt is [[-0.         -0.57735027 -0.57735027 -0.57735027]
 [ 0.          0.81649658 -0.40824829 -0.40824829]
 [ 0.          0.         -0.70710678  0.70710678]
 [ 1.          0.          0.          0.        ]]
For Matrix 4
Matrix U is [[-0.57735027  0.          0.         -0.57735027 -0.577
35027]
 [-0.57735027  0.          0.         -0.21132487  0.78867513]
 [-0.57735027  0.          0.          0.78867513 -0.21132487]
 [ 0.         -0.70710678 -0.70710678  0.          0.        ]
 [ 0.         -0.70710678  0.70710678  0.          0.        ]]
Matrix S is [[3. 0. 0. 0. 0.]
 [0. 2. 0. 0. 0.]
```

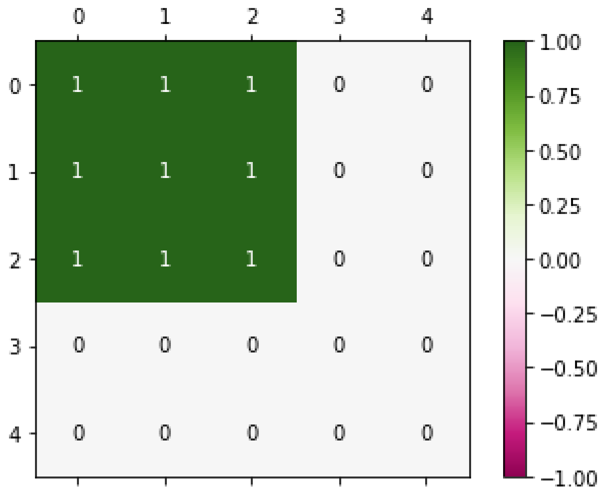
```
 [0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]]
Matrix Vt is [[-0.57735027 -0.57735027 -0.57735027 -0.         -0.
]
 [-0.         -0.         -0.         -0.70710678 -0.70710678]
 [ 0.          0.          0.         -0.70710678  0.70710678]
 [ 0.         -0.70710678  0.70710678  0.          0.        ]
 [ 0.81649658 -0.40824829 -0.40824829  0.          0.        ]]
For Matrix 5
Matrix U is [[-3.94102719e-01 -5.00000000e-01  3.07706105e-01  7.071
06781e-01
   8.41763023e-17]
 [-3.94102719e-01 -5.00000000e-01  3.07706105e-01 -7.07106781e-01
  -8.66774470e-17]
 [-6.15412209e-01 -2.77555756e-16 -7.88205438e-01  0.00000000e+00
   2.50114466e-18]
 [-3.94102719e-01  5.00000000e-01  3.07706105e-01  0.00000000e+00
  -7.07106781e-01]
 [-3.94102719e-01  5.00000000e-01  3.07706105e-01  1.11022302e-16
   7.07106781e-01]]
Matrix S is [[3.56155281e+00 0.00000000e+00 0.00000000e+00 0.0000000
0e+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 3.02510438e-17
  0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
  0.00000000e+00]]
Matrix Vt is [[-3.94102719e-01 -3.94102719e-01 -6.15412209e-01 -3.94
102719e-01
  -3.94102719e-01]
 [-5.00000000e-01 -5.00000000e-01 -1.94289029e-16  5.00000000e-01
   5.00000000e-01]
 [-3.07706105e-01 -3.07706105e-01  7.88205438e-01 -3.07706105e-01
  -3.07706105e-01]
 [-7.07106781e-01  7.07106781e-01  2.22044605e-16 -1.38777878e-16
  -8.32667268e-17]
 [ 0.00000000e+00 -2.31872909e-17 -9.30950307e-18 -7.07106781e-01
   7.07106781e-01]]
For Matrix 6
Matrix U is [[-4.61939766e-01 -1.91341716e-01  8.36419811e-01  2.245
03673e-01
   0.00000000e+00]
 [-4.61939766e-01 -1.91341716e-01 -4.90470696e-01  7.13749603e-01
   4.80660718e-17]
 [-3.82683432e-01  9.23879533e-01  2.22044605e-16 -5.55111512e-17
  -1.39805270e-17]
 [-4.61939766e-01 -1.91341716e-01 -1.72974557e-01 -4.69126638e-01
  -7.07106781e-01]
 [-4.61939766e-01 -1.91341716e-01 -1.72974557e-01 -4.69126638e-01
   7.07106781e-01]]
Matrix S is [[4.82842712e+00 0.00000000e+00 0.00000000e+00 0.0000000
0e+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 2.43075238e-16 0.00000000e+00
```

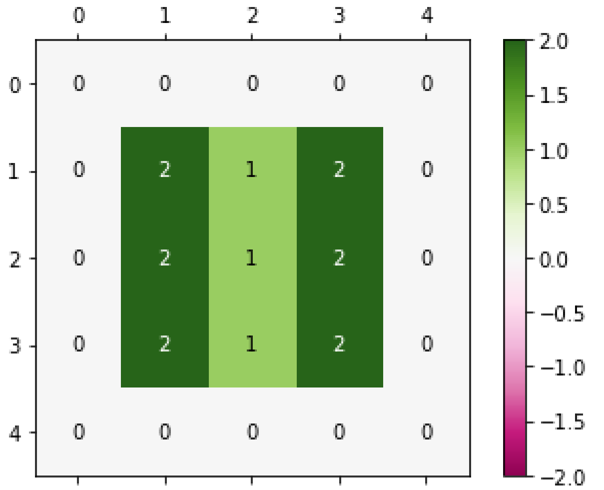
```
  0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 2.99007148e-18
  0.00000000e+00]
 [0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
  2.13821177e-50]]
Matrix Vt is [[-4.61939766e-01 -4.61939766e-01 -3.82683432e-01 -4.61
939766e-01
  -4.61939766e-01]
 [ 1.91341716e-01  1.91341716e-01 -9.23879533e-01  1.91341716e-01
   1.91341716e-01]
 [ 8.64514113e-01 -3.36387070e-01  1.11022302e-16 -2.64063522e-01
  -2.64063522e-01]
 [ 5.11404717e-02  7.98024899e-01 -8.32667268e-17 -4.24582685e-01
  -4.24582685e-01]
 [-0.00000000e+00 -4.23034501e-17  1.57626165e-17  7.07106781e-01
  -7.07106781e-01]]
```

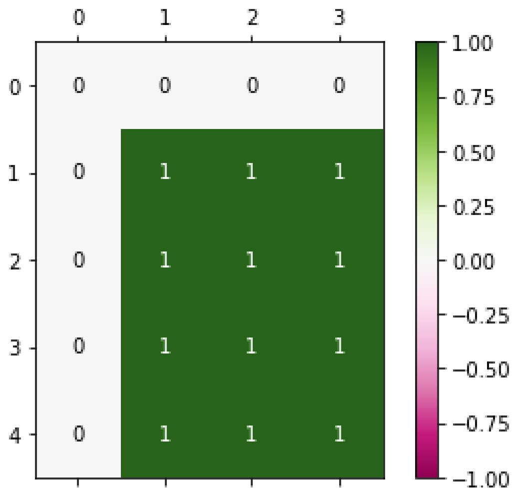
# 1c

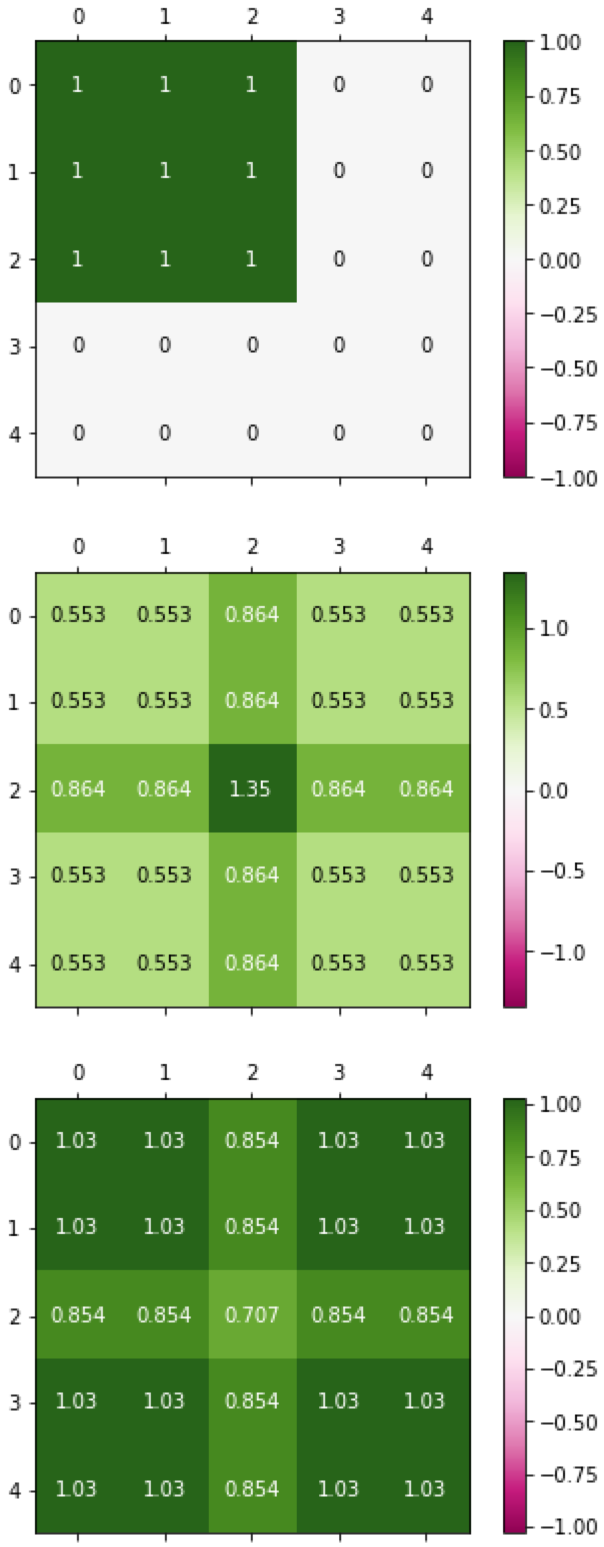
In [23]:

```
# You can use the functions svdcomp and plot_matrix from util.py
# YOUR PART
for matrix in [M1,M2,M3,M4,M5,M6]:
    A_1 = svdcomp(matrix, range(1))
    #print(A_1)
    plot_matrix(A_1)
```







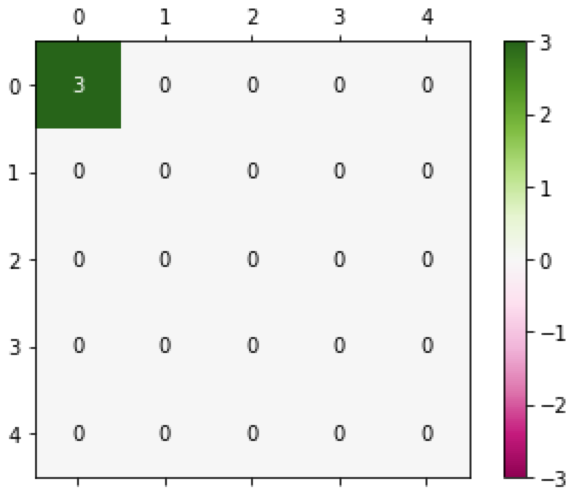

0 1 2 3 4
1 1 1 0 0
1 1 1 0 0
1 1 1 0 0
0 0 0 0 0
0 0 0 0 0
1.00 0.75 0.50 0.25 0.00 −0.25 −0.50 −0.75 −1.00
0.553 0.553 0.864 0.553 0.553
0.553 0.553 0.864 0.553 0.553
0.864 0.864 1.35 0.864 0.864
0.553 0.553 0.864 0.553 0.553
0.553 0.553 0.864 0.553 0.553
1.0 0.5 0.0 −0.5 −1.0
1.03 1.03 0.854 1.03 1.03
1.03 1.03 0.854 1.03 1.03
0.854 0.854 0.707 0.854 0.854
1.03 1.03 0.854 1.03 1.03
1.03 1.03 0.854 1.03 1.03
1.00 0.75 0.50 0.25 0.00 −0.25 −0.50 −0.75 −1.00

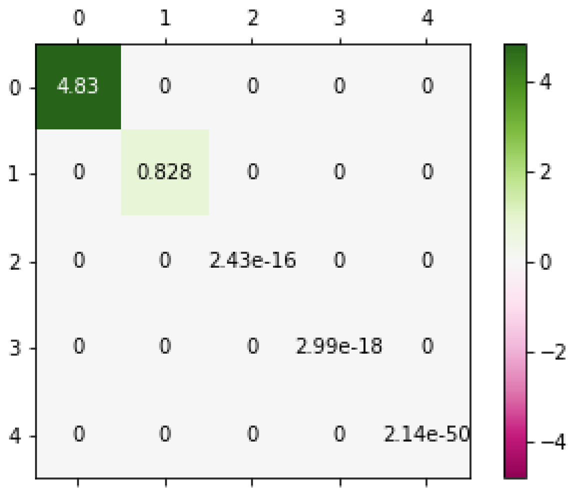
# 1d

In [24]:

```
# Another method to compute the rank is matrix_rank.
# YOUR PART
for matrix in [M1,M6]:
    U, S, Vt = compute_svd(matrix)
    r = matrix_rank(matrix)
    print(r)
    plot_matrix(S)
```

1
2




# 2 The SVD on Weather Data

In [7]:

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

In [30]:

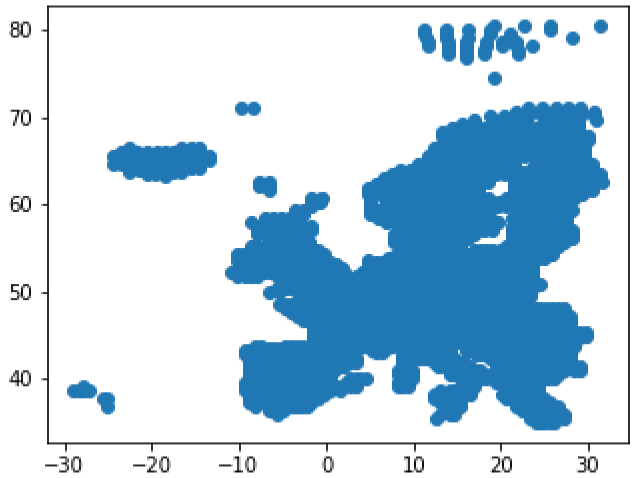
```
climate.head()
print(climate.describe())
```

```
              min1         min2         min3         min4          mi
n5  \
count  2575.000000  2575.000000  2575.000000  2575.000000  2575.0000
00
mean     -4.756272    -4.363223    -1.795767     1.642913     5.7999
22
std       6.491773     6.483893     5.842684     5.026565     4.3365
53
min     -23.200000   -23.600000   -22.500000   -19.600000   -11.4000
00
25%      -8.400000    -7.900000    -5.000000    -0.700000     3.7000
00
50%      -4.300000    -3.500000    -0.800000     2.600000     6.5000
00
75%      -0.100000     0.300000     2.100000     4.800000     8.3000
00
max      11.900000    11.300000    11.800000    13.300000    16.4000
00

              min6         min7         min8         min9          min
10  ...  \
count  2575.000000  2575.000000  2575.000000  2575.000000  2575.0000
00  ...
mean      9.571806    11.632777    11.285592     8.427573     4.5797
28  ...
std       4.006211     3.988869     4.185861     4.499939     4.7103
05  ...
min      -5.500000    -2.100000    -2.600000    -6.700000   -12.8000
00  ...
25%       7.700000     9.700000     9.100000     6.000000     2.2000
00  ...
50%      10.100000    11.900000    11.500000     8.600000     4.9000
00  ...
75%      11.700000    13.500000    13.400000    10.800000     7.0000
00  ...
max      20.900000    23.100000    23.800000    22.200000    19.0000
00  ...

             rain3        rain4        rain5        rain6          rai
n7  \
count  2575.000000  2575.000000  2575.000000  2575.000000  2575.0000
00
mean     55.710086    52.645918    58.542852    62.315994    61.3271
83
std      27.480366    20.281795    22.000305    27.374408    30.7188
71
min      18.500000    14.407000     7.500000     1.666700     0.0000
00
25%      34.122000    38.000000    42.211000    47.500000    45.2250
00
50%      48.833000    48.500000    55.917000    60.750000    65.1180
00
75%      68.958500    62.778000    72.436500    76.833000    79.5415
00
max     188.110000   141.170000   158.330000   181.170000   173.7500
00

             rain8        rain9       rain10       rain11         rain
12
count  2575.000000  2575.000000  2575.000000  2575.000000  2575.0000
00
```

```
mean      64.672127    66.604329    72.852245    76.256430    74.4000
58
std       30.665694    31.046708    36.411792    34.732055    38.6675
94
min        0.000000     8.250000    19.517000    20.815000    20.0000
00
25%       49.958500    48.308000    48.333000    51.700000    47.4170
00
50%       67.800000    62.083000    62.800000    66.600000    62.6250
00
75%       80.207000    76.933500    87.833000    93.050000    90.6040
00
max      186.670000   278.320000   310.710000   268.890000   302.0400
00

[8 rows x 48 columns]
```

In [8]:

```
# Plot the coordinates
plot_xy(lon, lat)
```




## 2a

In [9]:

```
# YOUR PART
# Center the data (i.e., substract the column mean from each column). Store the
 result
# in X.
print(climate.head())
print(climate.shape)
#X = np.zeros([2575,48])
X = pd.DataFrame()
for feature in climate:
    X[feature] = (climate[feature] - np.mean(climate[feature]))/np.std(climate[f
eature])

print(X.head())
```

```
   min1  min2  min3  min4  min5  min6  min7  min8  min9  min10  ...
rain3  \
0  10.6   9.9  10.5  11.0  12.5  14.7  16.9  18.0  17.1   15.0  ...
103.00
1   8.3   7.6   8.1   8.6  10.2  12.4  14.7  15.9  14.8   12.7  ...
108.29
2  10.1   9.5   9.9  10.5  12.0  14.2  16.4  17.4  16.6   14.6  ...
119.00
3  10.2   9.7   9.9  10.7  12.2  14.3  16.4  17.4  16.6   14.7  ...
141.20
4  11.7  11.1  11.5  12.1  13.5  15.7  17.8  18.9  18.1   16.1  ...
119.50

    rain4   rain5   rain6   rain7   rain8   rain9  rain10  rain11  r
ain12
0  74.000  66.000  53.000  41.000  57.000  92.000   118.0  126.00
126.0
1  72.429  63.286  50.571  38.857  53.714  88.571   118.0  125.43
120.0
2  75.000  60.400  48.400  38.400  50.400  84.400   120.4  127.00
117.8
3  80.400  54.200  44.000  36.600  42.600  75.600   123.6  129.40
114.4
4  74.000  58.500  47.500  38.000  48.500  83.500   121.0  125.00
116.0

[5 rows x 48 columns]
(2575, 48)
       min1      min2      min3      min4      min5      min6      m
in7  \
0  2.365957  2.200220  2.104881  1.861889  1.545324  1.280310  1.320
737
1  2.011593  1.845426  1.694031  1.384333  1.014846  0.706090  0.769
095
2  2.288921  2.138517  2.002169  1.762398  1.430003  1.155479  1.195
364
3  2.304328  2.169369  2.002169  1.802194  1.476131  1.180445  1.195
364
4  2.535435  2.385330  2.276069  2.080768  1.775967  1.529971  1.546
408

       min8      min9     min10  ...     rain3     rain4     rain5
rain6  \
0  1.604380  1.927607  2.212659  ...  1.721196  1.053074  0.339022 -
0.340384
1  1.102594  1.416389  1.724273  ...  1.913735  0.975600  0.215637 -
0.429133
2  1.461012  1.816472  2.127722  ...  2.303543  1.102389  0.084431 -
0.508457
3  1.461012  1.816472  2.148956  ...  3.111550  1.368689 -0.197438 -
0.669222
4  1.819431  2.149875  2.446234  ...  2.321741  1.053074 -0.001948 -
0.541340

      rain7     rain8     rain9    rain10    rain11    rain12
0 -0.661845 -0.250235  0.818142  1.240162  1.432487  1.334708
1 -0.731620 -0.357411  0.707674  1.240162  1.416073  1.179510
2 -0.746500 -0.465501  0.573302  1.306088  1.461285  1.122603
3 -0.805107 -0.719906  0.289803  1.393988  1.530399  1.034657
4 -0.759524 -0.527471  0.544307  1.322569  1.403690  1.076044
```

```
[5 rows x 48 columns]
```

In [10]:

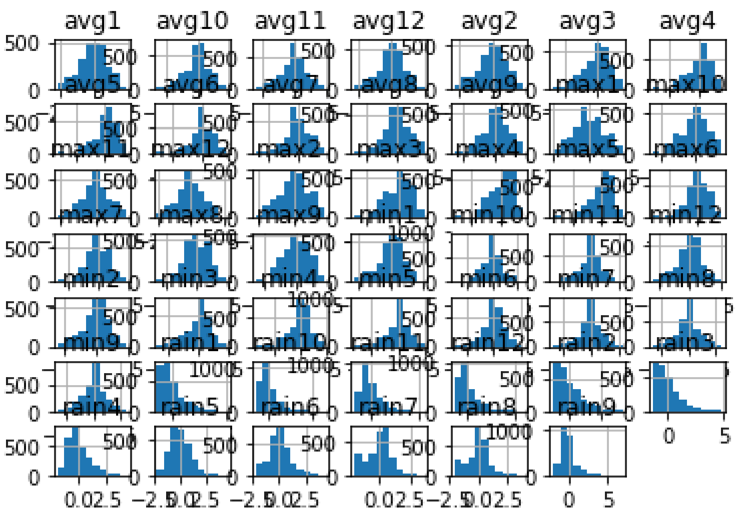
```
# Plot histograms of attributes
#nextplot()
X.hist(ax=plt.gca())
```

```
/Users/soumya/anaconda3/lib/python3.6/site-packages/IPython/core/int
eractiveshell.py:2963: UserWarning: To output multiple subplots, the
figure containing the passed axes is being cleared
  exec(code_obj, self.user_global_ns, self.user_ns)
```

Out[10]:

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x120e45860
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x1217364a8
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x12175ea20
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x10aa59be0
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x120e455f8
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x120e455c0
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x11eb5ce80
>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x11eabd748
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x129e2b9b0
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x11eb89080
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x11eaf9710
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x11eaa7da0
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x11eab9470
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x11eb39b70
>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x11eb582e8
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x11ecd1828
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x11ec5deb8
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x112a48588
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x112a70c18
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x129e4d2e8
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x129e72978
>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x129ea6048
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x129ecd6d8
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x129ef8d68
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x129f28438
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x129f4fac8
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x129f82198
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x129fa8828
>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x129fd1eb8
>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x12a004588
```

```
>,
         <matplotlib.axes._subplots.AxesSubplot object at 0x12a02bc18
>,
         <matplotlib.axes._subplots.AxesSubplot object at 0x12a05f2e8
>,
         <matplotlib.axes._subplots.AxesSubplot object at 0x12a086978
>,
         <matplotlib.axes._subplots.AxesSubplot object at 0x12a0b6048
>,
         <matplotlib.axes._subplots.AxesSubplot object at 0x12a0df6d8
>],
        [<matplotlib.axes._subplots.AxesSubplot object at 0x12a107d68
>,
         <matplotlib.axes._subplots.AxesSubplot object at 0x12a139438
>,
         <matplotlib.axes._subplots.AxesSubplot object at 0x12a160ac8
>,
         <matplotlib.axes._subplots.AxesSubplot object at 0x12a194198
>,
         <matplotlib.axes._subplots.AxesSubplot object at 0x12a1ba828
>,
         <matplotlib.axes._subplots.AxesSubplot object at 0x12a1e4eb8
>,
         <matplotlib.axes._subplots.AxesSubplot object at 0x12a216588
>],
        [<matplotlib.axes._subplots.AxesSubplot object at 0x12a23dc18
>,
         <matplotlib.axes._subplots.AxesSubplot object at 0x12a26e2e8
>,
         <matplotlib.axes._subplots.AxesSubplot object at 0x12a296978
>,
         <matplotlib.axes._subplots.AxesSubplot object at 0x12a2c9048
>,
         <matplotlib.axes._subplots.AxesSubplot object at 0x12a2f06d8
>,
         <matplotlib.axes._subplots.AxesSubplot object at 0x12a31bd68
>,
         <matplotlib.axes._subplots.AxesSubplot object at 0x12a34d438
>]],
      dtype=object)
```




# 2b

In [34]:

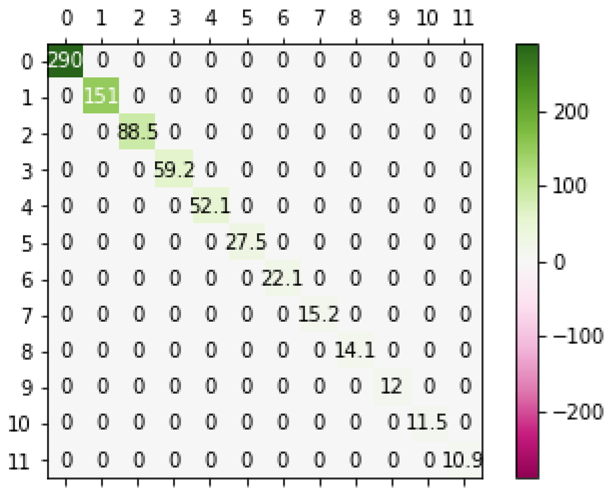
```
# Compute the SVD of the normalized climate data and store it in variables U,s,V
t. What
# is the rank of the data?
# YOUR PART
U, S, Vt = compute_svd(X)
r_x = matrix_rank(X)
print(r_x)
```

48

In [73]:

```
nextplot()
plot_matrix(S[:12,:12])
plt.savefig('2b_1')
```

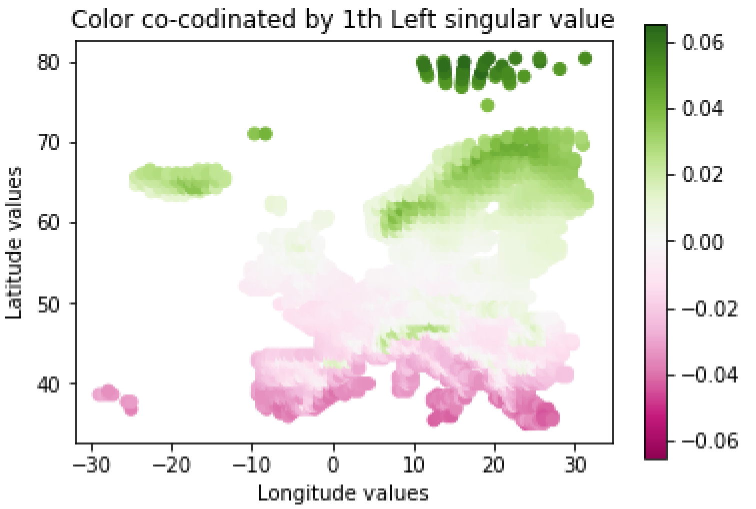
<Figure size 432x288 with 0 Axes>

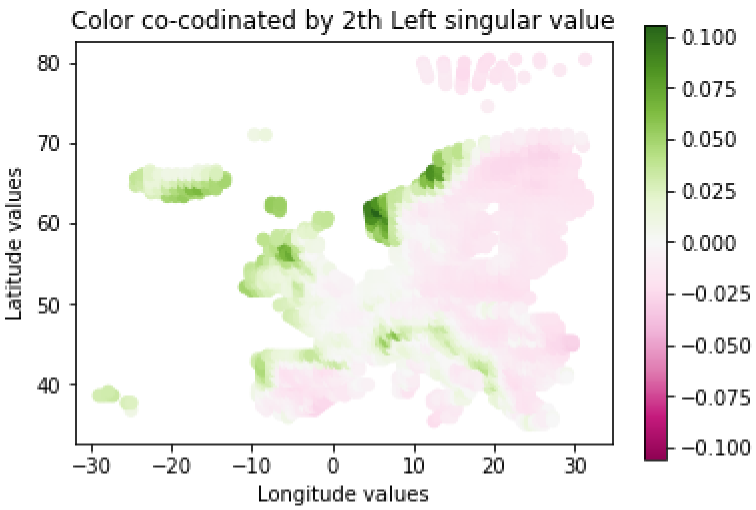


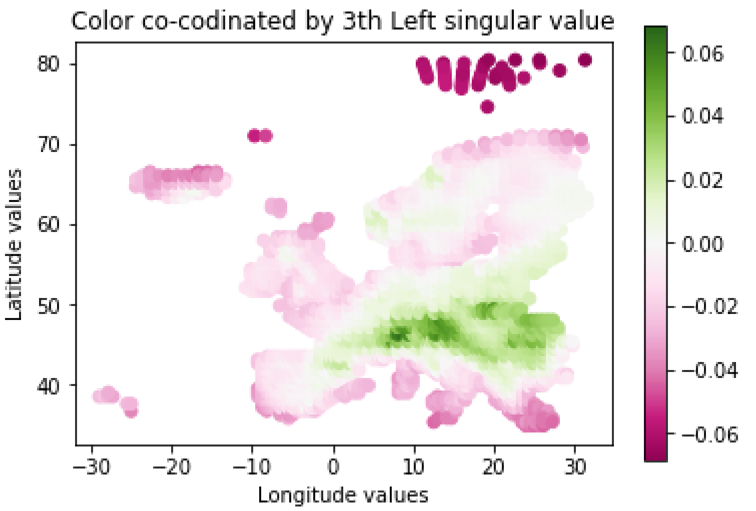

## 2c

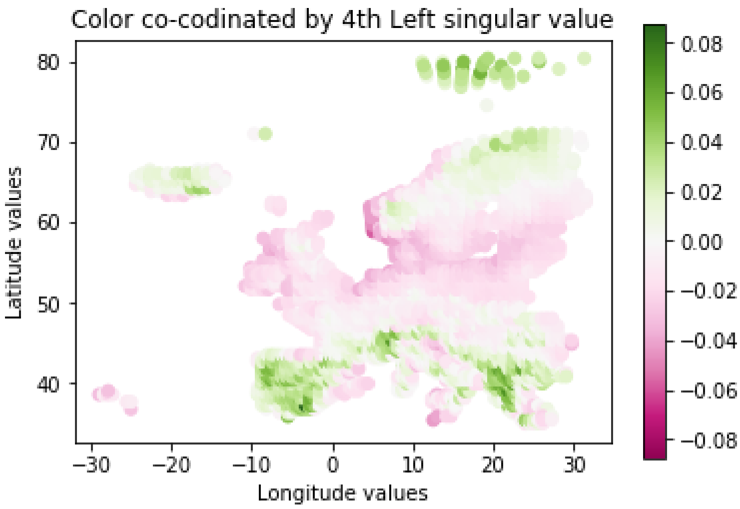
In [68]:

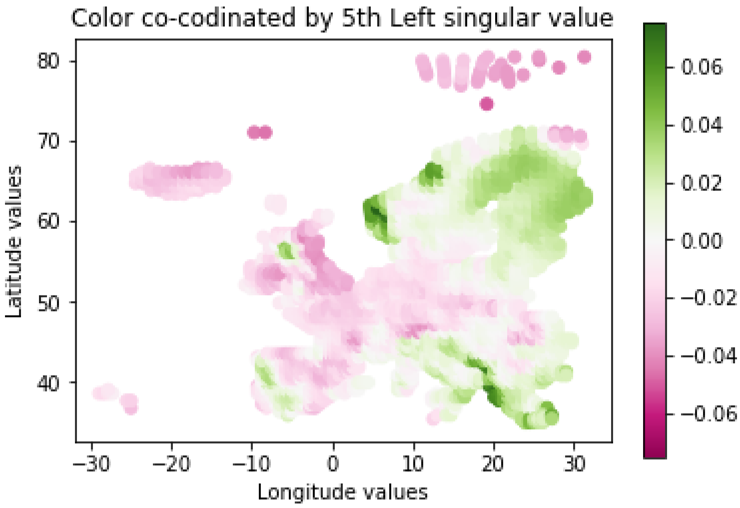
```
# Here is an example plot.
for i in [0,1,2,3,4]:
    plot_xy(lon, lat, U[:, i])
    plt.xlabel('Longitude values')
    plt.ylabel('Latitude values')
    plt.title('Color co-codinated by {}th Left singular value'.format(i+1))
    plt.savefig('2c_{}'.format(i))
```


Color co-codinated by 1th Left singular value
Latitude values
Longitude values
0.06
0.04
0.02
0.00
−0.02
−0.04
−0.06


Color co-codinated by 2th Left singular value
Latitude values
Longitude values
0.100
0.075
0.050
0.025
0.000
−0.025
−0.050
−0.075
−0.100

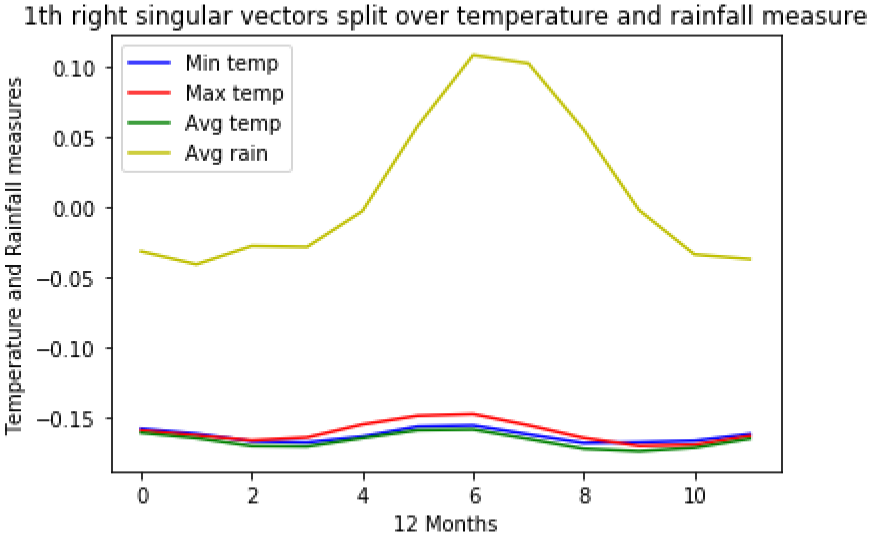

Color co-codinated by 3th Left singular value
Latitude values
Longitude values
0.06
0.04
0.02
0.00
−0.02
−0.04
−0.06

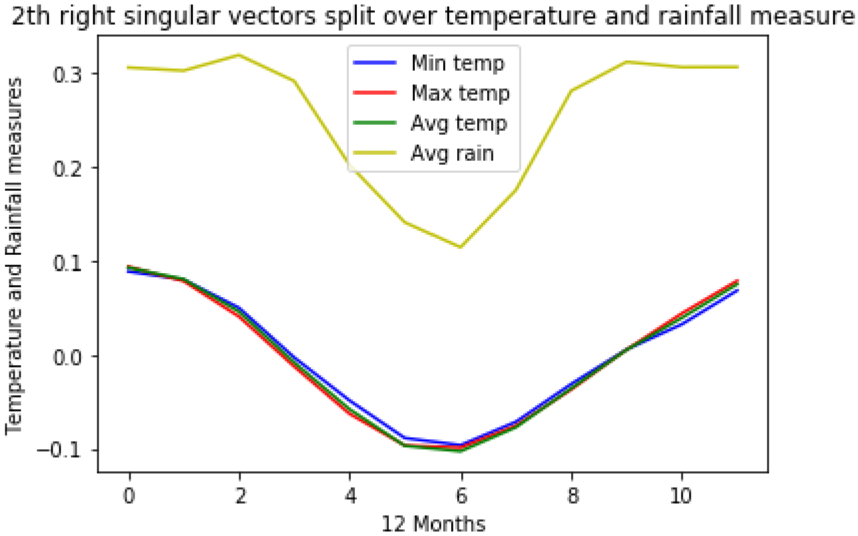

Color co-codinated by 4th Left singular value
Latitude values
Longitude values
0.08
0.06
0.04
0.02
0.00
−0.02
−0.04
−0.06
−0.08

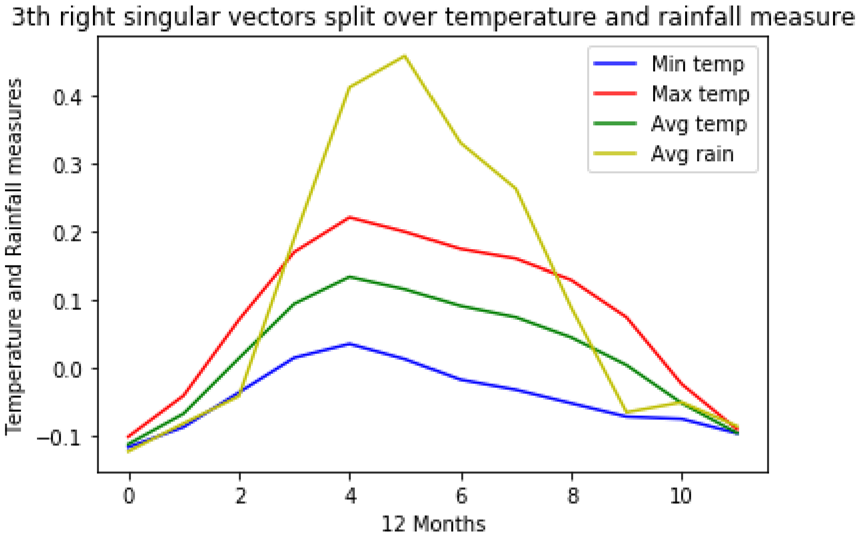

Color co-codinated by 5th Left singular value
Latitude values
Longitude values
0.06
0.04
0.02
0.00
−0.02
−0.04
−0.06

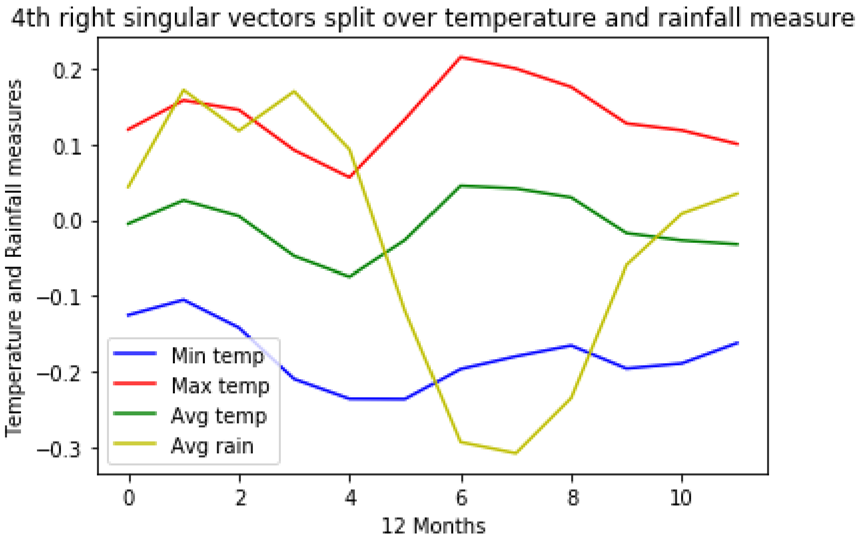
In [72]:

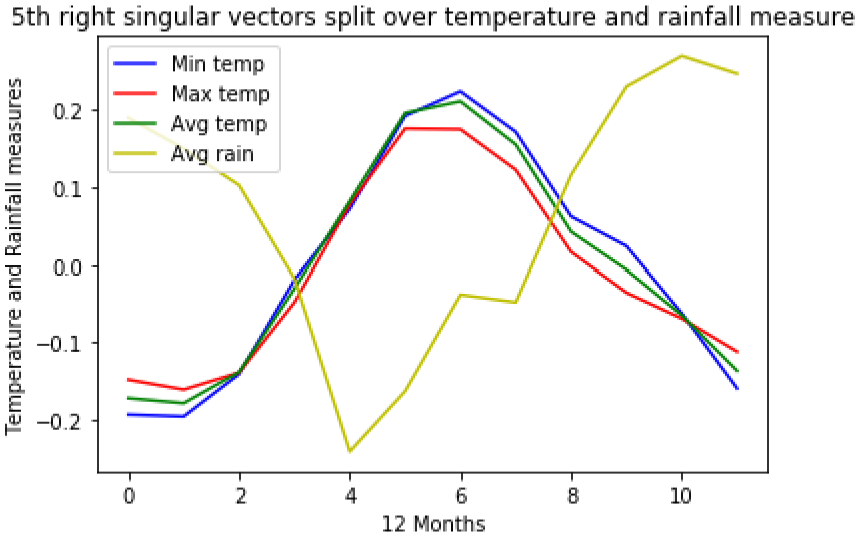
```
# For interpretation, it may also help to look at the other component matrices a
nd
# perhaps use other plot functions (e.g., plot_matrix).
# YOUR PART

#nextplot()
# plot_matrix(Vt[:1,:12])

# plt.plot(Vt[0, :12])

line_type = ['b-','r-','g-','y-']
labels = ['Min temp','Max temp','Avg temp','Avg rain']
range1 = [0,12,24,36]
range2 = [12,24,36,48]

for i in range(5):
    for ran1, ran2, line, label1 in zip(range1, range2, line_type, labels):
        plt.plot(range(12), Vt[i, ran1:ran2], line, label=label1)

    plt.title('{}th right singular vectors split over temperature and rainfall m
easure'.format(i+1))
    plt.ylabel('Temperature and Rainfall measures')
    plt.xlabel('12 Months')
    plt.legend()
    plt.savefig('2c2_{}'.format(i))

    plt.show()
    i=i+1
```


1th right singular vectors split over temperature and rainfall measure
Min temp
Max temp
Avg temp
Avg rain
Temperature and Rainfall measures
12 Months
0.10
0.05
0.00
−0.05
−0.10
−0.15
0
2
4
6
8
10


2th right singular vectors split over temperature and rainfall measure
Min temp
Max temp
Avg temp
Avg rain
Temperature and Rainfall measures
12 Months
0.3
0.2
0.1
0.0
−0.1
0
2
4
6
8
10


3th right singular vectors split over temperature and rainfall measure
Min temp
Max temp
Avg temp
Avg rain
Temperature and Rainfall measures
12 Months
0.4
0.3
0.2
0.1
0.0
−0.1
0
2
4
6
8
10

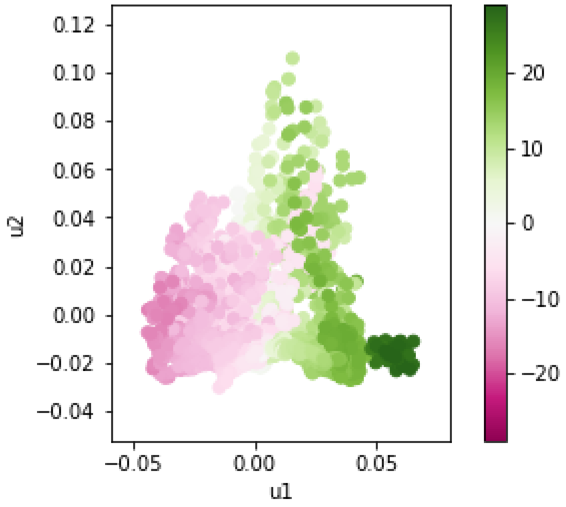

4th right singular vectors split over temperature and rainfall measure
Min temp
Max temp
Avg temp
Avg rain
Temperature and Rainfall measures
12 Months
0.2
0.1
0.0
−0.1
−0.2
−0.3
0
2
4
6
8
10

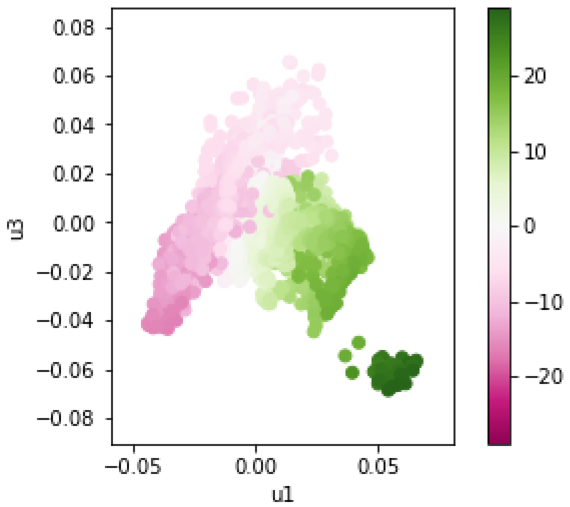

5th right singular vectors split over temperature and rainfall measure
Min temp
Max temp
Avg temp
Avg rain
Temperature and Rainfall measures
12 Months
0.2
0.1
0.0
−0.1
−0.2
0
2
4
6
8
10

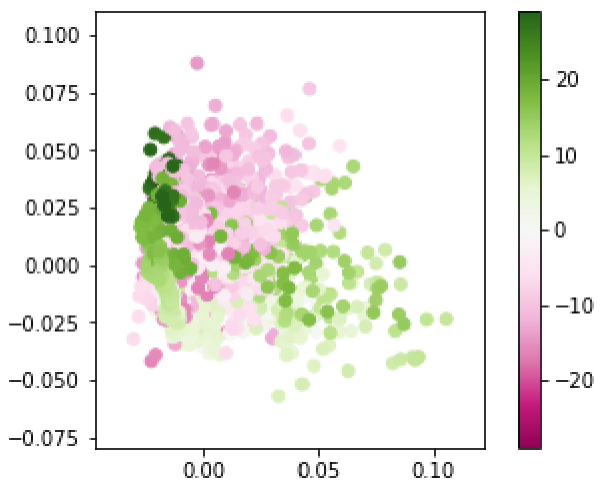
## 2d

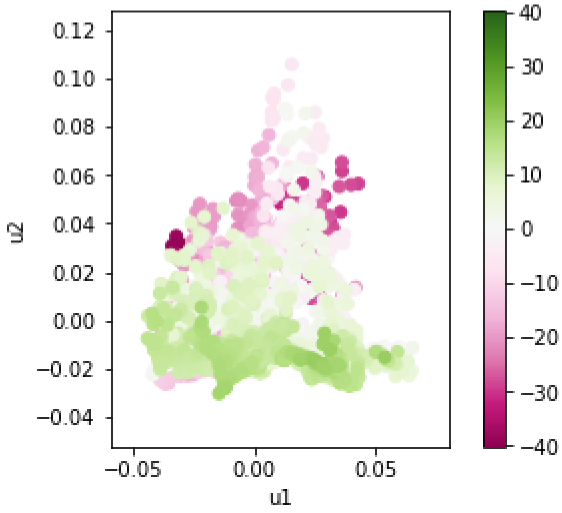
In [84]:

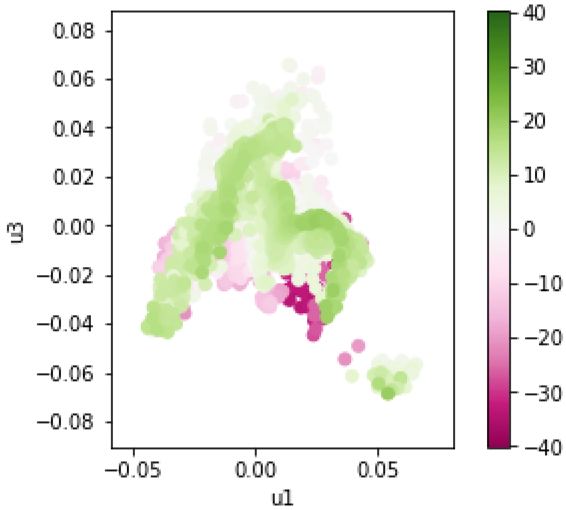
```
# Here is an example.
plot_xy(U[:, 0], U[:, 1], lat - np.mean(lat))
plt.ylabel('u2')
plt.xlabel('u1')
plt.savefig('2d1')
plot_xy(U[:, 0], U[:, 2], lat - np.mean(lat))
plt.ylabel('u3')
plt.xlabel('u1')
plt.savefig('2d2')
plot_xy(U[:, 1], U[:, 3], lat - np.mean(lat))


plot_xy(U[:, 0], U[:, 1], lon - np.mean(lon))
plt.ylabel('u2')
plt.xlabel('u1')
plt.savefig('2d3')
plot_xy(U[:, 0], U[:, 2], lon - np.mean(lon))
plt.ylabel('u3')
plt.xlabel('u1')
plt.savefig('2d4')
plot_xy(U[:, 1], U[:, 2], lon - np.mean(lon))
```

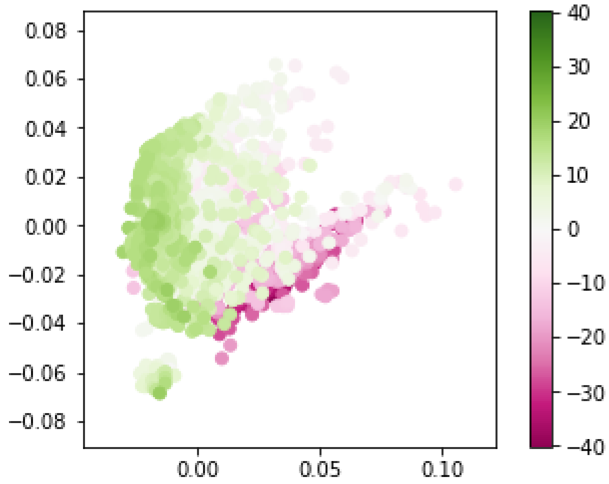

0.12
0.10
0.08
0.06
0.04
0.02
0.00
−0.02
−0.04
u2
−0.05
0.00
0.05
u1
20
10
0
−10
−20


0.08
0.06
0.04
0.02
0.00
−0.02
−0.04
−0.06
−0.08
u3
−0.05
0.00
0.05
u1
20
10
0
−10
−20


0.100
0.075
0.050
0.025
0.000
−0.025
−0.050
−0.075
0.00
0.05
0.10
20
10
0
−10
−20


u2
u1
0.12
0.10
0.08
0.06
0.04
0.02
0.00
−0.02
−0.04
−0.05
0.00
0.05
40
30
20
10
0
−10
−20
−30
−40


u3
u1
0.08
0.06
0.04
0.02
0.00
−0.02
−0.04
−0.06
−0.08
−0.05
0.00
0.05
40
30
20
10
0
−10
−20
−30
−40


0.08
0.06
0.04
0.02
0.00
−0.02
−0.04
−0.06
−0.08
0.00
0.05
0.10
40
30
20
10
0
−10
−20
−30
−40

## 2e

In [89]:

```
# 2e(i) Guttman-Kaiser
# YOUR PART
s_diag = S.diagonal()
print(s_diag)
diag_greater_one = s_diag[s_diag > 1]
print(diag_greater_one.size)
```

```
[2.90222389e+02 1.50668824e+02 8.84936404e+01 5.91859882e+01
 5.21202132e+01 2.74621244e+01 2.21341436e+01 1.52406513e+01
 1.41321813e+01 1.20289877e+01 1.14767046e+01 1.09209834e+01
 9.14704009e+00 8.39692373e+00 7.93211636e+00 7.06774614e+00
 6.74240524e+00 6.51838587e+00 5.76805648e+00 5.39678641e+00
 5.06878890e+00 4.21038123e+00 3.88507570e+00 3.37992885e+00
 3.12011424e+00 2.88184606e+00 2.53346089e+00 2.48165895e+00
 2.32967767e+00 2.07730350e+00 1.90548668e+00 1.86392296e+00
 1.72330112e+00 1.60158454e+00 1.28336386e+00 1.12607554e+00
 1.04958416e+00 9.84527428e-01 8.39920385e-01 6.56703416e-01
 4.94031887e-01 4.13336481e-01 3.78242622e-01 3.47242119e-01
 3.20887328e-01 3.06204289e-01 3.00736590e-01 2.55298845e-01]
37
```

In [90]:

```
# 2e(ii) 90% squared Frobenius norm
# YOUR PART
sq_fro = np.sum(np.square(S))
print(sq_fro)
k_90fro = 0
for i in range(len(s_diag)):
    print(np.sum(np.square(s_diag[:i+1])))
    if np.sum(np.square(s_diag[:i+1])) >= 0.9*sq_fro:
        k_90fro = i+1
        break
k_90fro
```

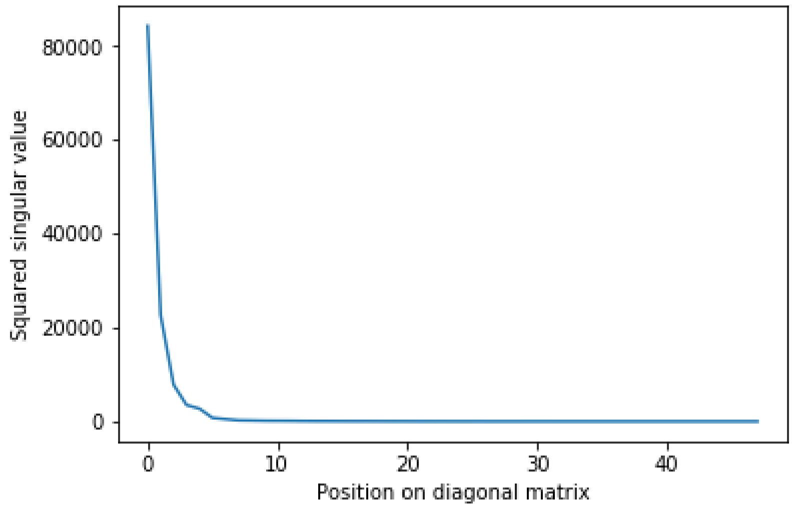
```
123600.00000000004
84229.03525535364
106930.12967573805
114761.25406592785
```

Out[90]:

```
3
```

In [124]:

```
# 2e(iii) Scree test
s_diag_sq = np.square(s_diag)
plt.plot(s_diag_sq)
plt.xlabel('Position on diagonal matrix')
plt.ylabel('Squared singular value')
plt.savefig('2eiii')
```




In [116]:

```
# 2e(iv) entropy
# YOUR PART
m,n = X.shape
fk = np.square(s_diag)/np.sum(np.square(s_diag))
log_fk = np.log(fk)
# log_fk

E = - (1/np.log(min(m,n))) * np.sum(fk[:min(m,n)] * log_fk[:min(m,n)])
print(E)

for i in range(len(fk)):
    print(np.sum(fk[:i+1]))
    if np.sum(fk[:i+1]) >= E:
        k_entropy = i+1
        break

print(k_entropy)
```

```
0.2752163447341984
0.6814646865319872
1
```

In [93]:

```
print(fk)
fk[:2]
```

```
[6.81464687e-01 1.83665812e-01 6.33586116e-02 2.83412718e-02
 2.19782898e-02 6.10168510e-03 3.96375659e-03 1.87926740e-03
 1.61584585e-03 1.17068403e-03 1.06565331e-03 9.64950480e-04
 6.76928337e-04 5.70455729e-04 5.09049110e-04 4.04150773e-04
 3.67799583e-04 3.43765003e-04 2.69178605e-04 2.35641614e-04
 2.07869101e-04 1.43424839e-04 1.22118230e-04 9.24265292e-05
 7.87630490e-05 6.71928535e-05 5.19289973e-05 4.98271128e-05
 4.39109874e-05 3.49125391e-05 2.93760477e-05 2.81084853e-05
 2.40272391e-05 2.07530182e-05 1.33254272e-05 1.02592729e-05
 8.91283902e-06 7.84218655e-06 5.70765576e-06 3.48915354e-06
 1.97465620e-06 1.38225766e-06 1.15750389e-06 9.75542795e-07
 8.33079911e-07 7.58584681e-07 7.31735409e-07 5.27326053e-07]
```

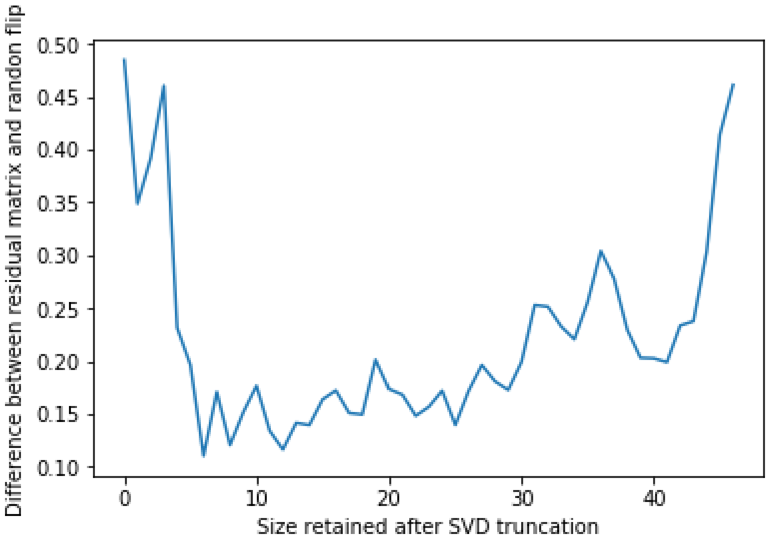
Out[93]:

```
array([0.68146469, 0.18366581])
```

In [125]:

```
# 2e(v) random flips
# Random sign matrix: np.random.choice([-1,1], X.shape)
# YOUR PART
diff = np.zeros(47)
for k_random in range(47):
    #k_random = 1
    Xk = svdcomp(X, components=range(k_random+1))
    X_k = X - Xk
    X_k_2 = np.linalg.norm(X_k, ord=2)
    X_k_cap = np.multiply(X_k, np.random.choice([-1,1], X.shape))
    X_k_2_cap = np.linalg.norm(X_k_cap, ord=2)

    diff[k_random] = (X_k_2 - X_k_2_cap) / np.linalg.norm(X_k, ord='fro')

print(diff)
plt.plot(diff)
plt.xlabel('Size retained after SVD truncation')
plt.ylabel('Difference between residual matrix and randon flip')
plt.savefig('2eiii')
```

```
[0.48482684 0.34873445 0.39166627 0.46047715 0.23150497 0.19635353
 0.11014896 0.17072515 0.12035587 0.1513948  0.17653529 0.13384191
 0.11611486 0.14137742 0.13929812 0.16362436 0.17212239 0.15076572
 0.14935492 0.20122029 0.17337331 0.16805344 0.14804853 0.15669474
 0.17189778 0.13937642 0.17169304 0.19626032 0.18076801 0.17255519
 0.19908758 0.25306621 0.25153811 0.23279639 0.22057749 0.25596704
 0.30374428 0.27808073 0.22956935 0.20299932 0.20259382 0.19891073
 0.23365995 0.23773744 0.30296771 0.41408024 0.46125479]
```



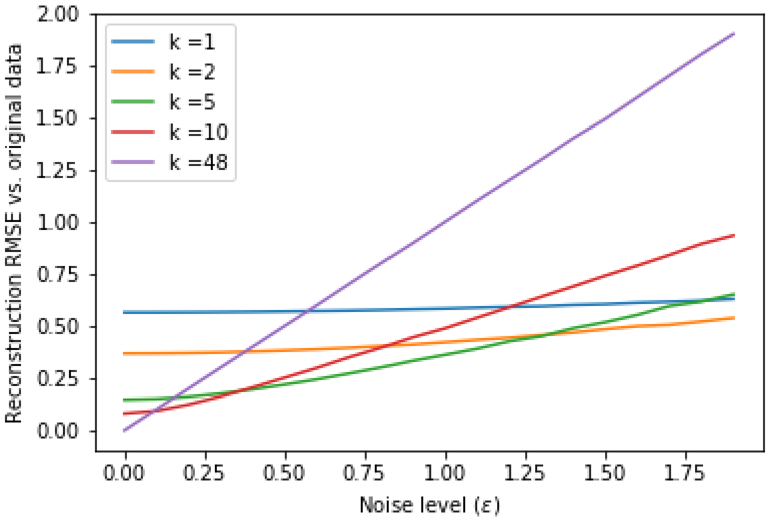
In [95]:

```
# 2e What, if any, of these would be your choice?
# YOUR PART
```

## 2f

In [100]:

```
# Here is the empty plot that you need to fill (one line per choice of k: RSME b
etween
# original X and the reconstruction from size-k SVD of noisy versions)
# YOUR PART

for k in [1,2,5,10,48]:
    i=0
    rmse = np.zeros(len(np.arange(0,2,0.1)))
    for epsilon in np.arange(0,2,0.1):
        X_noise = X + np.random.randn(*X.shape) * epsilon
        X_noise_recon = svdcomp(X_noise, components = range(k))
        rmse[i] = (1/np.sqrt(m*n)) * np.linalg.norm(X - X_noise_recon)
        i=i+1
    if i==0:
        nextplot()
    plt.plot(np.arange(0,2,0.1), rmse, label = 'k ={}'.format(k))

plt.xlabel(r"Noise level ($\epsilon$)")
plt.ylabel("Reconstruction RMSE vs. original data")
plt.legend()
plt.savefig('2f')
```




# 3 SVD and k-means

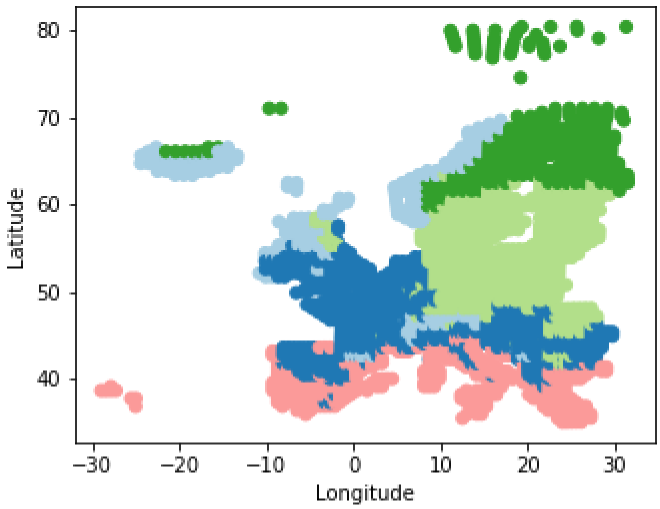
In [16]:

```
# 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_
```

# 3a

In [101]:

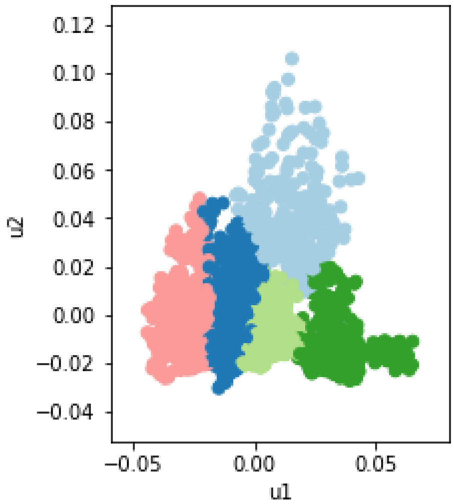
```
# Plot the results to the map: use the cluster labels to give the color to each
# point.
plot_xy(lon, lat, X_clusters)
plt.xlabel("Longitude")
plt.ylabel("Latitude")
plt.savefig('3a')
```




# 3b

In [102]:

```
# YOUR PART HERE
plot_xy(U[:, 0], U[:, 1], X_clusters)
plt.xlabel("u1")
plt.ylabel("u2")
plt.savefig('3b')
```




# 3c

In [122]:

```
# Compute the PCA scores, store in Z (of shape N x k)
k2 = 2
# YOUR PART HERE
U, S, Vt = compute_svd(X)
Z = svdcomp(X, components=range(k2))
Z.shape
Z = Z @ np.transpose(Vt[:k2, :])
#Z = np.dot(Z,np.transpose(Vt[1,:]))
print(Z)
```

```
[[-8.66502823  4.55581225]
 [-8.21979448  4.51199781]
 [-8.91586657  4.72061625]
 ...
 [ 8.2594551  -1.48025141]
 [ 9.13349919 -1.84214326]
 [ 8.42073364 -0.80435535]]
```

In [123]:

```
# cluster and visualize
for k in [1,2,3]:
    U, S, Vt = compute_svd(X)
    X_comp = svdcomp(X, components=range(k))
    Z = X_comp @ np.transpose(Vt[:k, :])
    Z_clusters = KMeans(5).fit(Z).labels_
    # match clusters as well as possible (try without)
    Z_clusters = match_categories(X_clusters, Z_clusters)
    nextplot()
    axs = plt.gcf().subplots(1, 2)
    plot_xy(lon, lat, X_clusters, axis=axs[0])
    plot_xy(lon, lat, Z_clusters, axis=axs[1])
    plt.savefig('3c{}'.format(k))
```