Multivariate

載入套件

In [11]:
```
!pip install prince
```

```
Collecting prince
  Obtaining dependency information for prince from https://files.pythonhosted.org/packages/a
1/ef/10313cfbb5479c0e693fed94c04d9ae1174331d907dfbfef7efd4ea75002/prince-0.12.1-py3-none-an
y.whl.metadata (https://files.pythonhosted.org/packages/a1/ef/10313cfbb5479c0e693fed94c04d9a
e1174331d907dfbfef7efd4ea75002/prince-0.12.1-py3-none-any.whl.metadata)
  Downloading prince-0.12.1-py3-none-any.whl.metadata (638 bytes)
Requirement already satisfied: altair<6.0.0,>=4.2.2 in /Users/penny/Desktop/pythonProject202
30716/venv/lib/python3.11/site-packages (from prince) (5.0.1)
Requirement already satisfied: pandas<3.0.0,>=1.4.1 in /Users/penny/Desktop/pythonProject202
30716/venv/lib/python3.11/site-packages (from prince) (2.0.3)
Requirement already satisfied: scikit-learn<2.0.0,>=1.0.2 in /Users/penny/Desktop/pythonProj
ect20230716/venv/lib/python3.11/site-packages (from prince) (1.3.0)
Requirement already satisfied: jinja2 in /Users/penny/Desktop/pythonProject20230716/venv/li
b/python3.11/site-packages (from altair<6.0.0,>=4.2.2->prince) (3.1.2)
Requirement already satisfied: jsonschema>=3.0 in /Users/penny/Desktop/pythonProject2023071
6/venv/lib/python3.11/site-packages (from altair<6.0.0,>=4.2.2->prince) (4.18.3)
Requirement already satisfied: numpy in /Users/penny/Desktop/pythonProject20230716/venv/lib/
python3.11/site-packages (from altair<6.0.0,>=4.2.2->prince) (1.24.3)
Requirement already satisfied: toolz in /Users/penny/Desktop/pythonProject20230716/venv/lib/
python3.11/site-packages (from altair<6.0.0,>=4.2.2->prince) (0.12.0)
Requirement already satisfied: python-dateutil>=2.8.2 in /Users/penny/Desktop/pythonProject2
0230716/venv/lib/python3.11/site-packages (from pandas<3.0.0,>=1.4.1->prince) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /Users/penny/Desktop/pythonProject20230716/ve
nv/lib/python3.11/site-packages (from pandas<3.0.0,>=1.4.1->prince) (2023.3)
Requirement already satisfied: tzdata>=2022.1 in /Users/penny/Desktop/pythonProject20230716/
venv/lib/python3.11/site-packages (from pandas<3.0.0,>=1.4.1->prince) (2023.3)
Requirement already satisfied: scipy>=1.5.0 in /Users/penny/Desktop/pythonProject20230716/ve
nv/lib/python3.11/site-packages (from scikit-learn<2.0.0,>=1.0.2->prince) (1.11.1)
Requirement already satisfied: joblib>=1.1.1 in /Users/penny/Desktop/pythonProject20230716/v
env/lib/python3.11/site-packages (from scikit-learn<2.0.0,>=1.0.2->prince) (1.3.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in /Users/penny/Desktop/pythonProject202
30716/venv/lib/python3.11/site-packages (from scikit-learn<2.0.0,>=1.0.2->prince) (3.2.0)
Requirement already satisfied: attrs>=22.2.0 in /Users/penny/Desktop/pythonProject20230716/v
env/lib/python3.11/site-packages (from jsonschema>=3.0->altair<6.0.0,>=4.2.2->prince) (23.1.
0)
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in /Users/penny/Desktop/
pythonProject20230716/venv/lib/python3.11/site-packages (from jsonschema>=3.0->altair<6.0.0,
>=4.2.2->prince) (2023.6.1)
Requirement already satisfied: referencing>=0.28.4 in /Users/penny/Desktop/pythonProject2023
0716/venv/lib/python3.11/site-packages (from jsonschema>=3.0->altair<6.0.0,>=4.2.2->prince)
(0.29.1)
Requirement already satisfied: rpds-py>=0.7.1 in /Users/penny/Desktop/pythonProject20230716/
venv/lib/python3.11/site-packages (from jsonschema>=3.0->altair<6.0.0,>=4.2.2->prince) (0.8.
10)
Requirement already satisfied: six>=1.5 in /Users/penny/Desktop/pythonProject20230716/venv/l
ib/python3.11/site-packages (from python-dateutil>=2.8.2->pandas<3.0.0,>=1.4.1->prince) (1.1
6.0)
Requirement already satisfied: MarkupSafe>=2.0 in /Users/penny/Desktop/pythonProject2023071
6/venv/lib/python3.11/site-packages (from jinja2->altair<6.0.0,>=4.2.2->prince) (2.1.3)
Downloading prince-0.12.1-py3-none-any.whl (415 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 415.1/415.1 kB 2.3 M
B/s eta 0:00:00a 0:00:01
Installing collected packages: prince
Successfully installed prince-0.12.1
```

In [347]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from sklearn.preprocessing import scale
import prince
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
```

載入檔案

In [13]:
```python
#載入資料集
df = pd.read_csv('./FOODP.csv')
print(f"The table contains: {df.shape[0]} rows and {df.shape[1]} columns")
```

```
The table contains: 24 rows and 6 columns
```

資料敘述統計

In [16]:
```python
df.head(24)
```

Out[16]:

	City	Bread	Hamburger	Butter	Apples	Tomatoes
0	Anchorage	70.9	135.6	155.00	63.9	100.1
1	Atlanta	36.4	111.5	144.30	53.9	95.9
2	Baltimore	28.9	108.8	151.00	47.5	104.5
3	Boston	43.2	119.3	142.00	41.1	96.5
4	Buffalo	34.5	109.9	124.80	35.6	75.9
5	Chicago	37.1	107.5	145.40	65.1	94.2
6	Cincinnati	37.1	118.1	149.60	45.6	90.8
7	Cleveland	38.5	107.7	142.70	50.3	83.2
8	Dallas	35.5	116.8	142.50	62.4	90.7
9	Detroit	40.8	108.8	140.10	39.7	96.1
10	Honolulu	50.9	131.7	154.40	65.0	93.9
11	Houston	35.1	102.3	150.30	59.3	84.5
12	Kansas City	35.1	99.8	162.30	42.6	87.9
13	Los Angeles	36.9	96.2	140.40	54.7	79.3
14	Milwaukee	33.3	109.1	123.20	57.7	87.7
15	Minneapolis	32.5	116.7	135.10	48.0	89.1
16	New York	42.7	130.8	148.70	47.6	92.1
17	Philadelphia	42.9	126.9	153.80	51.9	101.5
18	Pittsburgh	36.9	115.4	138.90	43.8	91.9
19	St. Louis	36.9	109.8	140.00	46.7	79.0
20	San Diego	32.5	84.5	145.90	48.5	82.3
21	San Francisco	40.0	104.6	139.10	59.2	81.9
22	Seattle	32.2	105.4	136.80	54.0	88.6
23	Washington	31.8	116.7	154.81	57.6	86.6

In [14]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 24 entries, 0 to 23
Data columns (total 6 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   City       24 non-null     object 
 1   Bread      24 non-null     float64
 2   Hamburger  24 non-null     float64
 3   Butter     24 non-null     float64
 4   Apples     24 non-null     float64
 5   Tomatoes   24 non-null     float64
dtypes: float64(5), object(1)
memory usage: 1.3+ KB
```

In [15]:
```
#確認遺漏值
# any missing values
df.isnull().sum()
```

Out[15]:
```
City         0
Bread        0
Hamburger    0
Butter       0
Apples       0
Tomatoes     0
dtype: int64
```

In [18]:
```
#分類變數，將數值變數抓出
numerical_features = ['Bread', 'Hamburger', 'Butter',
                      'Apples', 'Tomatoes']

df[numerical_features].describe().round(2).T #.T將橫縱軸轉置
```

Out[18]:

	count	mean	std	min	25%	50%	75%	max
Bread	24.0	38.44	8.37	28.9	34.20	36.90	40.20	70.9
Hamburger	24.0	112.25	11.63	84.5	106.98	109.85	117.12	135.6
Butter	24.0	144.21	9.23	123.2	139.78	143.50	150.48	162.3
Apples	24.0	51.74	8.36	35.6	46.42	51.10	58.08	65.1
Tomatoes	24.0	89.76	7.40	75.9	84.18	89.90	94.62	104.5

In [176]:
```
df_city= df['City']
df_city
```

Out[176]:
```
0          Anchorage
1            Atlanta
2          Baltimore
3             Boston
4            Buffalo
5            Chicago
6         Cincinnati
7          Cleveland
8             Dallas
9            Detroit
10          Honolulu
11           Houston
12       Kansas City
13       Los Angeles
14         Milwaukee
15       Minneapolis
16          New York
17      Philadelphia
18        Pittsburgh
19         St. Louis
20         San Diego
21     San Francisco
22           Seattle
23        Washington
Name: City, dtype: object
```

In [262]:
```
df[numerical_features]
```

Out[262]:

	Bread	Hamburger	Butter	Apples	Tomatoes
0	70.9	135.6	155.00	63.9	100.1
1	36.4	111.5	144.30	53.9	95.9
2	28.9	108.8	151.00	47.5	104.5
3	43.2	119.3	142.00	41.1	96.5
4	34.5	109.9	124.80	35.6	75.9
5	37.1	107.5	145.40	65.1	94.2
6	37.1	118.1	149.60	45.6	90.8
7	38.5	107.7	142.70	50.3	83.2
8	35.5	116.8	142.50	62.4	90.7
9	40.8	108.8	140.10	39.7	96.1
10	50.9	131.7	154.40	65.0	93.9
11	35.1	102.3	150.30	59.3	84.5
12	35.1	99.8	162.30	42.6	87.9
13	36.9	96.2	140.40	54.7	79.3
14	33.3	109.1	123.20	57.7	87.7
15	32.5	116.7	135.10	48.0	89.1
16	42.7	130.8	148.70	47.6	92.1
17	42.9	126.9	153.80	51.9	101.5
18	36.9	115.4	138.90	43.8	91.9
19	36.9	109.8	140.00	46.7	79.0
20	32.5	84.5	145.90	48.5	82.3
21	40.0	104.6	139.10	59.2	81.9
22	32.2	105.4	136.80	54.0	88.6
23	31.8	116.7	154.81	57.6	86.6

In [273]:
```
x1_Bread= np.array(df['Bread'])
x1_Bread
```

Out[273]:
```
array([70.9, 36.4, 28.9, 43.2, 34.5, 37.1, 37.1, 38.5, 35.5, 40.8, 50.9,
       35.1, 35.1, 36.9, 33.3, 32.5, 42.7, 42.9, 36.9, 36.9, 32.5, 40. ,
       32.2, 31.8])
```

In [274]:
```
x2_Hamburger=np.array(df['Hamburger'])
x2_Hamburger
```

Out[274]:
```
array([135.6, 111.5, 108.8, 119.3, 109.9, 107.5, 118.1, 107.7, 116.8,
       108.8, 131.7, 102.3,  99.8,  96.2, 109.1, 116.7, 130.8, 126.9,
       115.4, 109.8,  84.5, 104.6, 105.4, 116.7])
```

In [275]:
```
x3_Butter=np.array(df['Butter'])
x3_Butter
```

Out[275]:
```
array([155.  , 144.3 , 151.  , 142.  , 124.8 , 145.4 , 149.6 , 142.7 ,
       142.5 , 140.1 , 154.4 , 150.3 , 162.3 , 140.4 , 123.2 , 135.1 ,
       148.7 , 153.8 , 138.9 , 140.  , 145.9 , 139.1 , 136.8 , 154.81])
```

In [276]:
```
x4_Apples=np.array(df['Apples'])
x4_Apples
```

Out[276]:
```
array([63.9, 53.9, 47.5, 41.1, 35.6, 65.1, 45.6, 50.3, 62.4, 39.7, 65. ,
       59.3, 42.6, 54.7, 57.7, 48. , 47.6, 51.9, 43.8, 46.7, 48.5, 59.2,
       54. , 57.6])
```

In [277]:
```python
x5_Tomatoes=np.array(df['Tomatoes'])
x5_Tomatoes
```

Out[277]:
```
array([100.1,  95.9, 104.5,  96.5,  75.9,  94.2,  90.8,  83.2,  90.7,
        96.1,  93.9,  84.5,  87.9,  79.3,  87.7,  89.1,  92.1, 101.5,
        91.9,  79. ,  82.3,  81.9,  88.6,  86.6])
```

In [283]:
```python
x1_x5=np.array([x1_Bread,x2_Hamburger,x3_Butter,x4_Apples,x5_Tomatoes])
x1_x5
```

Out[283]:
```
array([[ 70.9 ,  36.4 ,  28.9 ,  43.2 ,  34.5 ,  37.1 ,  37.1 ,  38.5 ,
         35.5 ,  40.8 ,  50.9 ,  35.1 ,  35.1 ,  36.9 ,  33.3 ,  32.5 ,
         42.7 ,  42.9 ,  36.9 ,  36.9 ,  32.5 ,  40.  ,  32.2 ,  31.8 ],
       [135.6 , 111.5 , 108.8 , 119.3 , 109.9 , 107.5 , 118.1 , 107.7 ,
        116.8 , 108.8 , 131.7 , 102.3 ,  99.8 ,  96.2 , 109.1 , 116.7 ,
        130.8 , 126.9 , 115.4 , 109.8 ,  84.5 , 104.6 , 105.4 , 116.7 ],
       [155.  , 144.3 , 151.  , 142.  , 124.8 , 145.4 , 149.6 , 142.7 ,
        142.5 , 140.1 , 154.4 , 150.3 , 162.3 , 140.4 , 123.2 , 135.1 ,
        148.7 , 153.8 , 138.9 , 140.  , 145.9 , 139.1 , 136.8 , 154.81],
       [ 63.9 ,  53.9 ,  47.5 ,  41.1 ,  35.6 ,  65.1 ,  45.6 ,  50.3 ,
         62.4 ,  39.7 ,  65.  ,  59.3 ,  42.6 ,  54.7 ,  57.7 ,  48.  ,
         47.6 ,  51.9 ,  43.8 ,  46.7 ,  48.5 ,  59.2 ,  54.  ,  57.6 ],
       [100.1 ,  95.9 , 104.5 ,  96.5 ,  75.9 ,  94.2 ,  90.8 ,  83.2 ,
         90.7 ,  96.1 ,  93.9 ,  84.5 ,  87.9 ,  79.3 ,  87.7 ,  89.1 ,
         92.1 , 101.5 ,  91.9 ,  79.  ,  82.3 ,  81.9 ,  88.6 ,  86.6 ]])
```

In [285]:
```python
#使用numpy
corr_matrix = np.corrcoef(x1_x5).round(decimals=4)
corr_matrix
```

Out[285]:
```
array([[1.    , 0.6491, 0.3302, 0.3187, 0.3621],
       [0.6491, 1.    , 0.2448, 0.1909, 0.5558],
       [0.3302, 0.2448, 1.    , 0.2351, 0.4361],
       [0.3187, 0.1909, 0.2351, 1.    , 0.1334],
       [0.3621, 0.5558, 0.4361, 0.1334, 1.    ]])
```

In [289]:
```python
#相關係數
corr_matrix = df[numerical_features].corr()    #pd.DataFrame.corr(df[numerical_features])
print(corr_matrix)
```

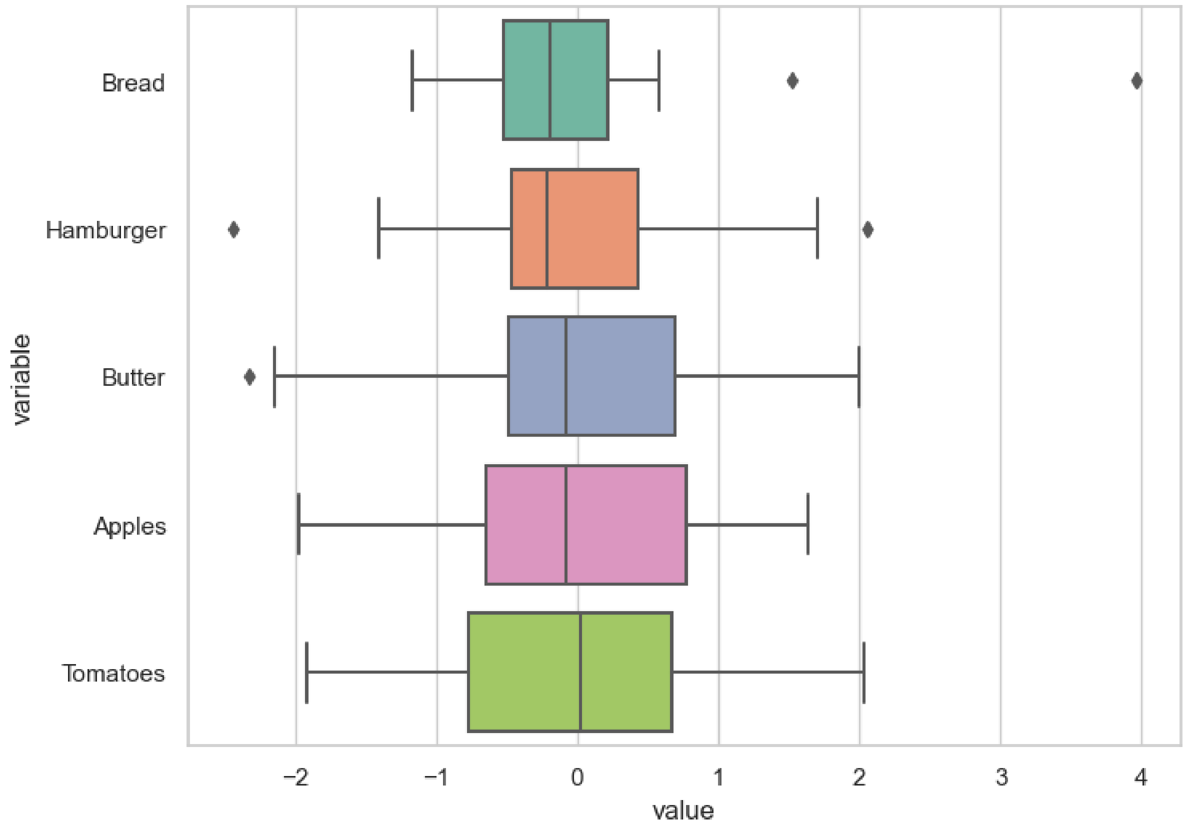
```
              Bread  Hamburger    Butter    Apples  Tomatoes
Bread      1.000000   0.649053  0.330177  0.318703  0.362068
Hamburger  0.649053   1.000000  0.244778  0.190896  0.555799
Butter     0.330177   0.244778  1.000000  0.235142  0.436129
Apples     0.318703   0.190896  0.235142  1.000000  0.133384
Tomatoes   0.362068   0.555799  0.436129  0.133384  1.000000
```

In [291]:
```python
#共變異矩陣
cov_matrix =  df[numerical_features].cov()              #pd.DataFrame.cov(df[numerical_f
print(cov_matrix)
```

```
               Bread   Hamburger     Butter     Apples   Tomatoes
Bread      69.999058   63.171920  25.488743  22.288804  22.413116
Hamburger  63.171920  135.330417  26.273947  18.562989  47.838949
Butter     25.488743   26.273947  85.135569  18.135969  29.773953
Apples     22.288804   18.562989  18.135973  69.872880   8.249457
Tomatoes   22.413116   47.838949  29.773953   8.249457  54.743406
```

In [21]:

```python
#將數值型變數資料做標準化
#使用scale()
df_scaled = scale(df[numerical_features])
df2 = pd.DataFrame(df_scaled, columns=numerical_features)
df2['City'] = pd.Series(df['City'], index=df.index)
df3 = pd.melt(df2, id_vars='City', value_vars=df2[numerical_features])
plt.figure(figsize=(8,6))
sns.set(style="whitegrid")
sns.boxplot(y='variable',x='value', data=df3, palette="Set2")
plt.show()
```



In [23]:

```python
df2[numerical_features].describe().loc[['count', 'mean','std']].round().T
```

Out[23]:

	count	mean	std
Bread	24.0	-0.0	1.0
Hamburger	24.0	-0.0	1.0
Butter	24.0	-0.0	1.0
Apples	24.0	-0.0	1.0
Tomatoes	24.0	0.0	1.0

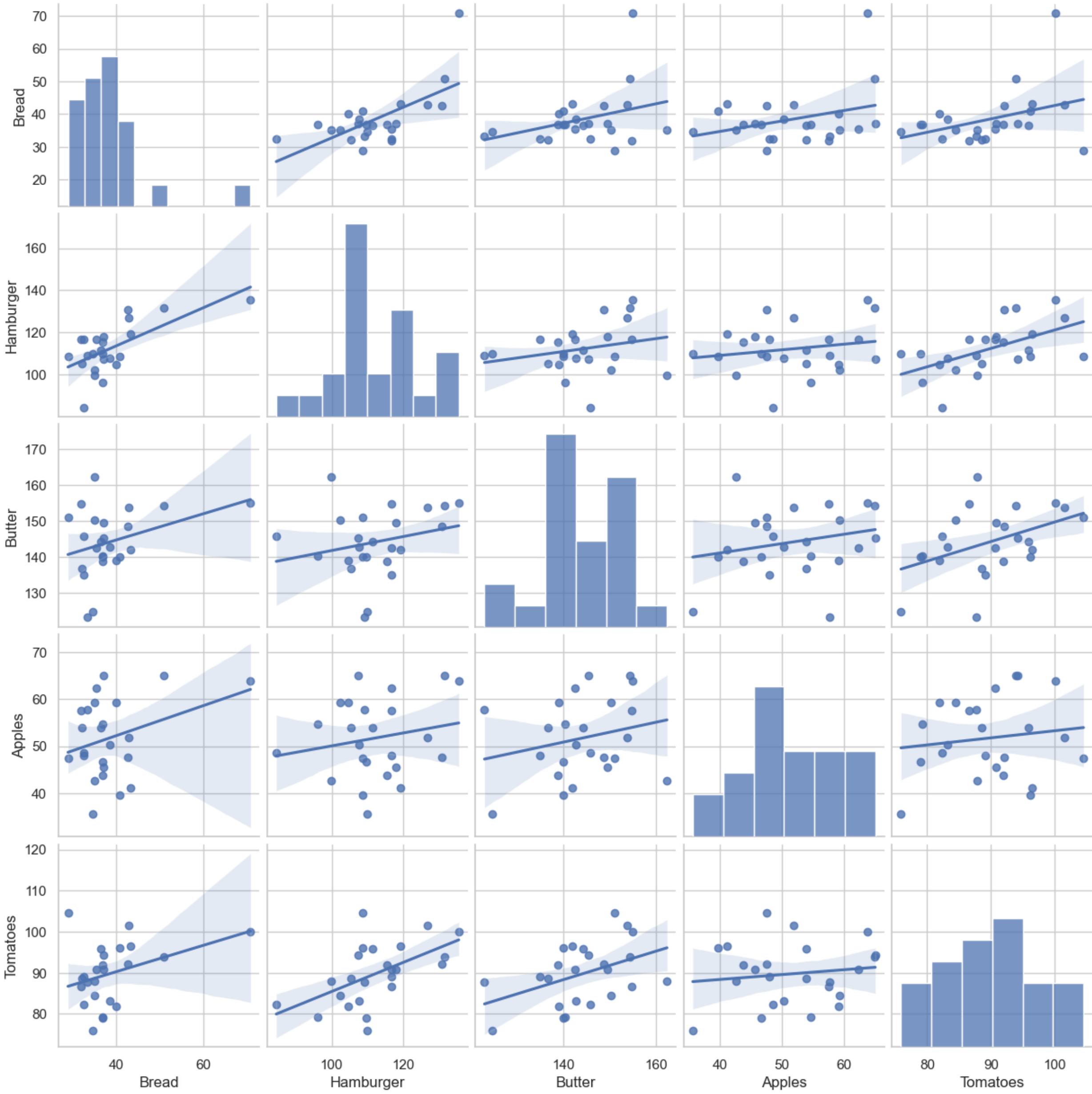
In [415]:
```
df2
```

Out[415]:

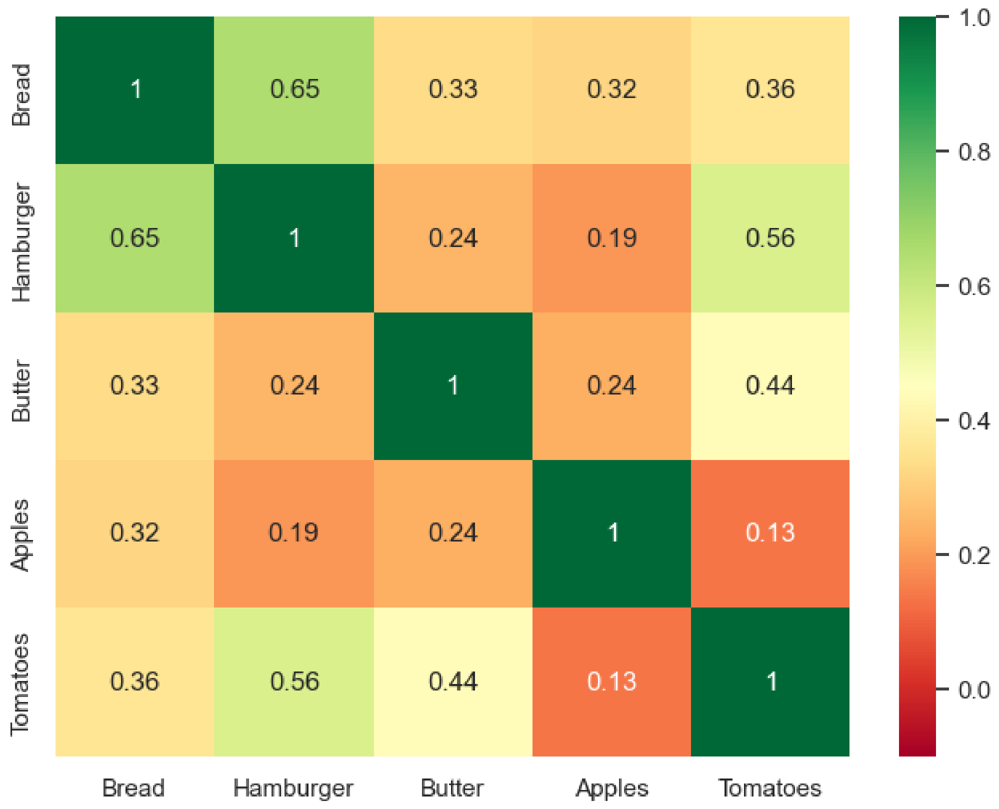
	Bread	Hamburger	Butter	Apples	Tomatoes	City
0	3.962979	2.050729	1.194236	1.486313	1.427797	Anchorage
1	-0.249276	-0.065492	0.009641	0.264267	0.847934	Atlanta
2	-1.164984	-0.302579	0.751397	-0.517842	2.035272	Baltimore
3	0.580966	0.619426	-0.244992	-1.299951	0.930771	Boston
4	-0.481255	-0.205988	-2.149201	-1.972076	-1.913316	Buffalo
5	-0.163810	-0.416732	0.131422	1.632958	0.613228	Chicago
6	-0.163810	0.514054	0.596403	-0.750030	0.143815	Cincinnati
7	0.007122	-0.399170	-0.167495	-0.175669	-0.905460	Cleveland
8	-0.359161	0.399901	-0.189637	1.303006	0.130009	Dallas
9	0.287939	-0.302579	-0.455340	-1.471037	0.875546	Detroit
10	1.521092	1.708270	1.127810	1.620738	0.571809	Honolulu
11	-0.407999	-0.873344	0.673900	0.924172	-0.725979	Houston
12	-0.407999	-1.092869	2.002418	-1.116644	-0.256566	Kansas City
13	-0.188229	-1.408985	-0.422127	0.362031	-1.443904	Los Angeles
14	-0.627769	-0.276236	-2.326336	0.728645	-0.284179	Milwaukee
15	-0.725444	0.391120	-1.008889	-0.456740	-0.090891	Minneapolis
16	0.519918	1.629241	0.496764	-0.505621	0.323296	New York
17	0.544337	1.286782	1.061384	0.019858	1.621084	Philadelphia
18	-0.188229	0.276967	-0.588192	-0.969999	0.295684	Pittsburgh
19	-0.188229	-0.214769	-0.466411	-0.615605	-1.485323	St. Louis
20	-0.725444	-2.436362	0.186777	-0.395637	-1.029716	San Diego
21	0.190264	-0.671381	-0.566050	0.911951	-1.084941	San Francisco
22	-0.762072	-0.601133	-0.820683	0.276488	-0.159922	Seattle
23	-0.810910	0.391120	1.173201	0.716424	-0.436047	Washington

In [26]:
```
#畫pairplot圖
#sns.set(style="whitegrid")
sns.pairplot(df[numerical_features], kind='reg', diag_kind='kde')
plt.show()
```

```
/Users/penny/Desktop/pythonProject20230716/venv/lib/python3.11/site-packages/seaborn/axisgri
d.py:118: UserWarning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```



In [27]:
```
#畫熱力圖
plt.figure(figsize=(8,6))
sns.set(style="whitegrid")
sns.heatmap(df2[numerical_features].corr(method='pearson'), vmin=-.1, vmax=1,  annot=True, cma
plt.show()
```



PCA

Mean-Corrected Data(不設定n_components)

In [486]:
```
#PCA對特徵降維
# 不設定參數
#PCA旨在找到讓特徵映射後資料變異量最大的投影向量，屬於無監督式學習，所以在這裡我們只要fit特徵x即可
# 不設定參數
pca1 = PCA()
pca_mean = pca1.fit_transform(df[numerical_features])
pca1.explained_variance_ratio_ #看新特徵的解釋能力
```

Out[486]:
```
array([0.52229379, 0.19063238, 0.15001509, 0.08352694, 0.0535318 ])
```

In [487]:
```
#看特徵數量
print(pca1.n_components_)
```

```
5
```

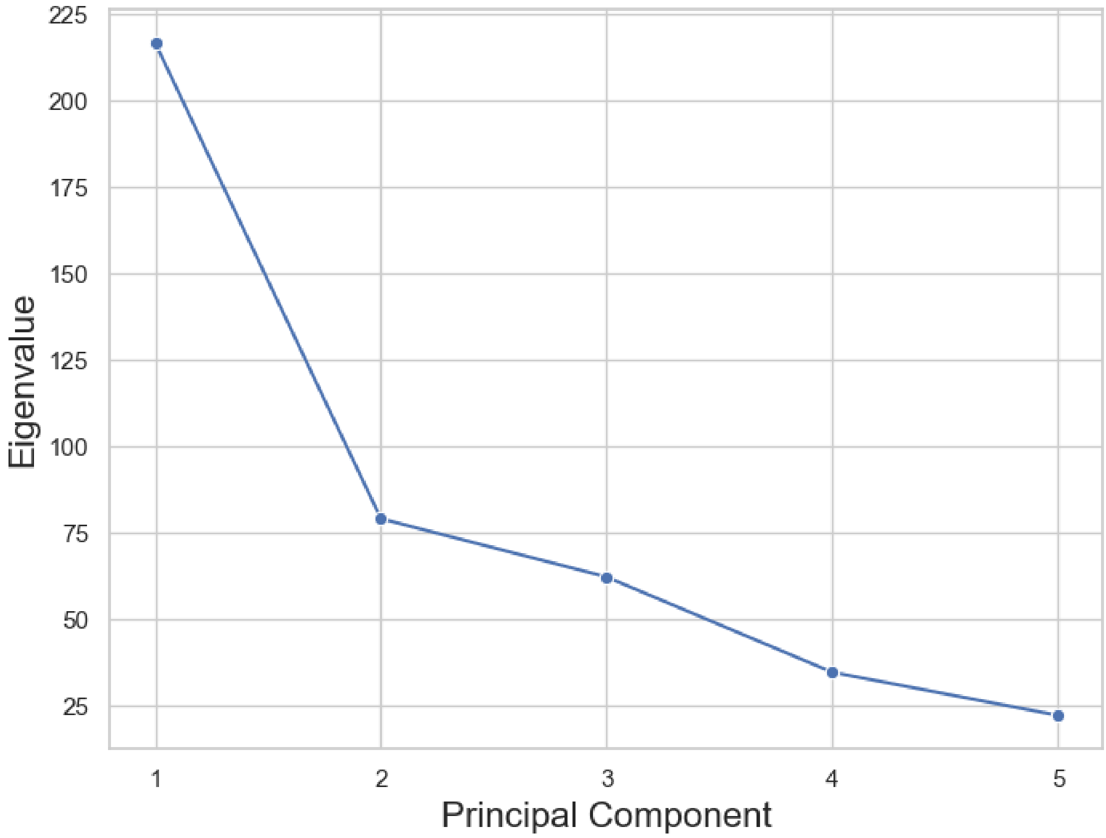
In [488]:
```
#印出5個特徵值(eigenvalue)
print(pca1.explained_variance_)
```

```
[216.79440165  79.12794127  62.26846303  34.67047386  22.22005045]
```

In [489]:
```
# 總變異
np.sum(pca1.explained_variance_)
```

Out[489]:
```
415.0813302536236
```

In [490]:
```
#印出陡坡圖
dset = pd.DataFrame()
dset['pca'] = range(1,6)
dset['eigenvalue'] = pd.DataFrame(pca1.explained_variance_)
plt.figure(figsize=(8,6))
sns.lineplot(x='pca', y='eigenvalue', marker="o", data=dset)
#使用matplotlib模組中plt.xticks()函數設定標線為整數
plt.xticks(np.arange(1, 6, 1))
plt.ylabel('Eigenvalue', fontsize=16)
plt.xlabel('Principal Component', fontsize=16)
plt.show()
```



In [491]:
```
# 加總可解釋變異
np.sum(pca1.explained_variance_ratio_)
```

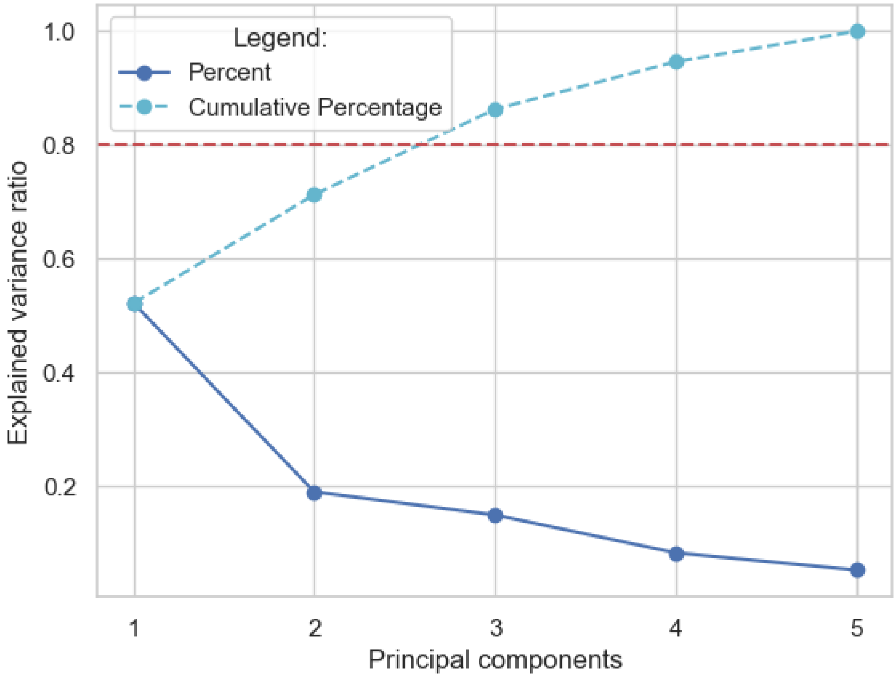
Out[491]: 0.9999999999999999

In [492]:
```
#印出可解釋百分比
print(pca1.explained_variance_ratio_)
```

```
[0.52229379 0.19063238 0.15001509 0.08352694 0.0535318 ]
```

In [493]:
```
# 繪製解釋變異數圖
plt.plot(range(1, 6), pca1.explained_variance_ratio_, 'o-', label='Percent')
plt.plot(range(1, 6), np.cumsum(pca1.explained_variance_ratio_),'o--', color='c', label='Cumul


plt.xticks(np.arange(1, 6, 1))
plt.ylabel('Explained variance ratio')
plt.xlabel('Principal components')
plt.legend(title='Legend:')
plt.axhline(0.8, color='r', linestyle='--');
```



In [494]:
```
#共變異數矩陣
pd.DataFrame(pca1.get_covariance())
```

Out[494]:

	0	1	2	3	4
0	69.999058	63.171920	25.488743	22.288804	22.413116
1	63.171920	135.330417	26.273947	18.562989	47.838949
2	25.488743	26.273947	85.135569	18.135973	29.773953
3	22.288804	18.562989	18.135973	69.872880	8.249457
4	22.413116	47.838949	29.773953	8.249457	54.743406

In [502]:
```
pca1.singular_values_
```

Out[502]:
```
array([70.61353438, 42.66078585, 37.84408342, 28.23864194, 22.60666186])
```

In [226]:
```
prin_name=pca1.get_feature_names_out()
prin_name
```

Out[226]:
```
array(['pca0', 'pca1', 'pca2', 'pca3', 'pca4'], dtype=object)
```

In [233]:
```
pca_mean_new = pd.DataFrame(pca_mean, index=df_city)
pca_mean_new
```

Out[233]:

City	0	1	2	3	4
Anchorage	41.320304	-0.867302	-8.270996	-11.054571	-8.453726
Atlanta	1.180461	-1.855410	0.704929	5.709833	-3.032091
Baltimore	-0.298380	-6.446398	13.154430	12.319367	-4.219468
Boston	6.442271	9.509017	9.023553	-1.226410	-4.588422
Buffalo	-18.413035	21.117024	0.984539	-7.401849	3.106132
Chicago	0.889789	-9.293280	-8.713408	7.064428	-3.321232
Cincinnati	4.412433	1.268550	8.076252	-0.712227	4.127236
Cleveland	-6.329041	-0.036316	-1.741145	-4.794888	1.398015
Dallas	4.018030	-1.282676	-8.362703	6.981982	3.229686
Detroit	-3.240609	6.328017	9.485600	-1.305549	-8.788818
Honolulu	27.361430	-3.496942	-7.247752	-1.152570	3.970371
Houston	-6.715949	-12.542950	-4.734735	-0.990727	2.480236
Kansas City	-6.932417	-15.886246	15.076000	-7.029182	1.740404
Los Angeles	-16.436863	-5.625248	-7.499938	-5.727723	-1.291204
Milwaukee	-11.104458	11.649573	-13.573917	7.934327	-3.381942
Minneapolis	-3.650685	10.377551	0.254098	4.863575	2.801885
New York	16.611905	7.500732	5.572972	-1.239309	6.349524
Philadelphia	19.855807	-0.679707	7.220762	4.189367	0.146156
Pittsburgh	-1.251636	8.738825	5.250807	1.264217	-1.019981
St. Louis	-8.719980	4.748177	-0.963898	-6.596042	4.919912
San Diego	-25.250773	-13.252416	1.573323	-5.545645	-5.256567
San Francisco	-7.576231	-2.475772	-11.344611	-3.439996	-0.639454
Seattle	-10.137220	1.032199	-4.239401	4.856563	-1.359565
Washington	3.964846	-8.529001	0.315238	3.033031	11.082914

In [192]:
```
pca_mean_new.index
```

Out[192]:
```
Index(['Anchorage', 'Atlanta', 'Baltimore', 'Boston', 'Buffalo', 'Chicago',
       'Cincinnati', 'Cleveland', 'Dallas', 'Detroit', 'Honolulu', 'Houston',
       'Kansas City', 'Los Angeles', 'Milwaukee', 'Minneapolis', 'New York',
       'Philadelphia', 'Pittsburgh', 'St. Louis', 'San Diego', 'San Francisco',
       'Seattle', 'Washington'],
      dtype='object', name='City')
```

In [208]:
```
pca_mean[:,1]
```

Out[208]:
```
array([ -0.8673024 ,  -1.85541048,  -6.44639844,   9.50901748,
        21.11702447,  -9.29328032,   1.26855005,  -0.03631648,
        -1.28267554,   6.3280169 ,  -3.49694233, -12.54294979,
       -15.88624639,  -5.62524792,  11.64957263,  10.37755132,
         7.50073245,  -0.67970723,   8.73882467,   4.74817704,
       -13.25241603,  -2.4757717 ,   1.03219903,  -8.52900099])
```

In [244]:
```
df.shape
```

Out[244]:
```
(24, 7)
```

In [ ]:

In [510]:
```
pca = prince.PCA(
     n_components=5,
  #  n_iter=10,
     rescale_with_mean=False,
     rescale_with_std=False,
     copy=True,
     check_input=True,
     engine='sklearn',
     random_state=234
 )
pca1_3 = pca.fit(df[numerical_features])
```

In [511]:
```
print(pca1_3.eigenvalues_)
```

```
[4.57847200e+04 1.00129688e+02 6.24987735e+01 3.56083085e+01
 2.24110105e+01]
```

In [512]:
```
#看特徵數量
pca1_3.eigenvalues_summary
```

Out[512]:

component	eigenvalue	% of variance	% of variance (cumulative)
0	45,784.720	99.52%	99.52%
1	100.130	0.22%	99.74%
2	62.499	0.14%	99.87%
3	35.608	0.08%	99.95%
4	22.411	0.05%	100.00%

In [ ]:
```

```

Mean-Corrected Data(n_components=2)

In [292]:
```
#PCA對特徵降維
# 設定參數=2
#PCA旨在找到讓特徵映射後資料變異量最大的投影向量，屬於無監督式學習，所以在這裡我們只要fit特徵x即可
# 不設定參數
pca1_1 = PCA(n_components=2)
pca_mean1_1 = pca1_1.fit_transform(df[numerical_features])
pca1_1.explained_variance_ratio_ #看新特徵的解釋能力
```

Out[292]:
```
array([0.52229379, 0.19063238])
```

In [293]:
```
#印出特徵值(eigenvalue)
print(pca1_1.explained_variance_)
```

```
[216.79440165  79.12794127]
```

In [294]:
```
# 總變異
np.sum(pca1_1.explained_variance_)
```

Out[294]:
```
295.9223429145531
```

In [295]:
```
# 加總可解釋變異
np.sum(pca1_1.explained_variance_ratio_)
```

Out[295]:
```
0.7129261697550651
```

In [297]:
```
#共變異數矩陣
pd.DataFrame(pca1_1.get_covariance())
```
Out[297]:

	0	1	2	3	4
0	76.162241	58.374218	25.556824	16.740652	27.704635
1	58.374218	139.501573	28.412604	19.658508	42.726291
2	25.556824	28.412604	82.632034	26.019730	22.723858
3	16.740652	19.658508	26.019730	55.569579	14.609465
4	27.704635	42.726291	22.723858	14.609465	61.215903

In [298]:
```
prin_name1_1=pca1_1.get_feature_names_out()
prin_name1_1
```
Out[298]:
```
array(['pca0', 'pca1'], dtype=object)
```

In [299]:
```
pca_mean_new1_1 = pd.DataFrame(pca_mean1_1, index=df_city)
pca_mean_new1_1
```
Out[299]:

	0	1
City		
Anchorage	41.320304	-0.867302
Atlanta	1.180461	-1.855410
Baltimore	-0.298380	-6.446398
Boston	6.442271	9.509017
Buffalo	-18.413035	21.117024
Chicago	0.889789	-9.293280
Cincinnati	4.412433	1.268550
Cleveland	-6.329041	-0.036316
Dallas	4.018030	-1.282676
Detroit	-3.240609	6.328017
Honolulu	27.361430	-3.496942
Houston	-6.715949	-12.542950
Kansas City	-6.932417	-15.886246
Los Angeles	-16.436863	-5.625248
Milwaukee	-11.104458	11.649573
Minneapolis	-3.650685	10.377551
New York	16.611905	7.500732
Philadelphia	19.855807	-0.679707
Pittsburgh	-1.251636	8.738825
St. Louis	-8.719980	4.748177
San Diego	-25.250773	-13.252416
San Francisco	-7.576231	-2.475772
Seattle	-10.137220	1.032199
Washington	3.964846	-8.529001

In [302]:
```
pca_mean_new1_1[0]
```

Out[302]:
```
City
Anchorage        41.320304
Atlanta           1.180461
Baltimore        -0.298380
Boston            6.442271
Buffalo         -18.413035
Chicago           0.889789
Cincinnati        4.412433
Cleveland        -6.329041
Dallas            4.018030
Detroit          -3.240609
Honolulu         27.361430
Houston          -6.715949
Kansas City      -6.932417
Los Angeles     -16.436863
Milwaukee       -11.104458
Minneapolis      -3.650685
New York         16.611905
Philadelphia     19.855807
Pittsburgh       -1.251636
St. Louis        -8.719980
San Diego       -25.250773
San Francisco    -7.576231
Seattle         -10.137220
Washington        3.964846
Name: 0, dtype: float64
```

In [311]:
```
x_print1= np.array(pca_mean_new1_1[0])
x_print1
```

Out[311]:
```
array([ 41.32030406,   1.1804606 ,  -0.29838036,   6.44227083,
       -18.41303452,   0.88978906,   4.41243322,  -6.329041  ,
         4.01802955,  -3.24060879,  27.36143016,  -6.71594941,
        -6.93241652, -16.43686343, -11.10445751,  -3.65068504,
        16.61190505,  19.85580739,  -1.25163574,  -8.71997999,
       -25.25077257,  -7.57623079, -10.1372203 ,   3.96484604])
```

In [317]:
```
type(x_print1)
```

Out[317]:
```
numpy.ndarray
```

In [312]:
```
y_print2= np.array(pca_mean_new1_1[1])
y_print2
```

Out[312]:
```
array([ -0.8673024 ,  -1.85541048,  -6.44639844,   9.50901748,
        21.11702447,  -9.29328032,   1.26855005,  -0.03631648,
        -1.28267554,   6.3280169 ,  -3.49694233, -12.54294979,
       -15.88624639,  -5.62524792,  11.64957263,  10.37755132,
         7.50073245,  -0.67970723,   8.73882467,   4.74817704,
       -13.25241603,  -2.4757717 ,   1.03219903,  -8.52900099])
```

In [315]:
```
test1=np.array(pca_mean_new1_1.index)
test1
```

Out[315]:
```
array(['Anchorage', 'Atlanta', 'Baltimore', 'Boston', 'Buffalo',
       'Chicago', 'Cincinnati', 'Cleveland', 'Dallas', 'Detroit',
       'Honolulu', 'Houston', 'Kansas City', 'Los Angeles', 'Milwaukee',
       'Minneapolis', 'New York', 'Philadelphia', 'Pittsburgh',
       'St. Louis', 'San Diego', 'San Francisco', 'Seattle', 'Washington'],
      dtype=object)
```

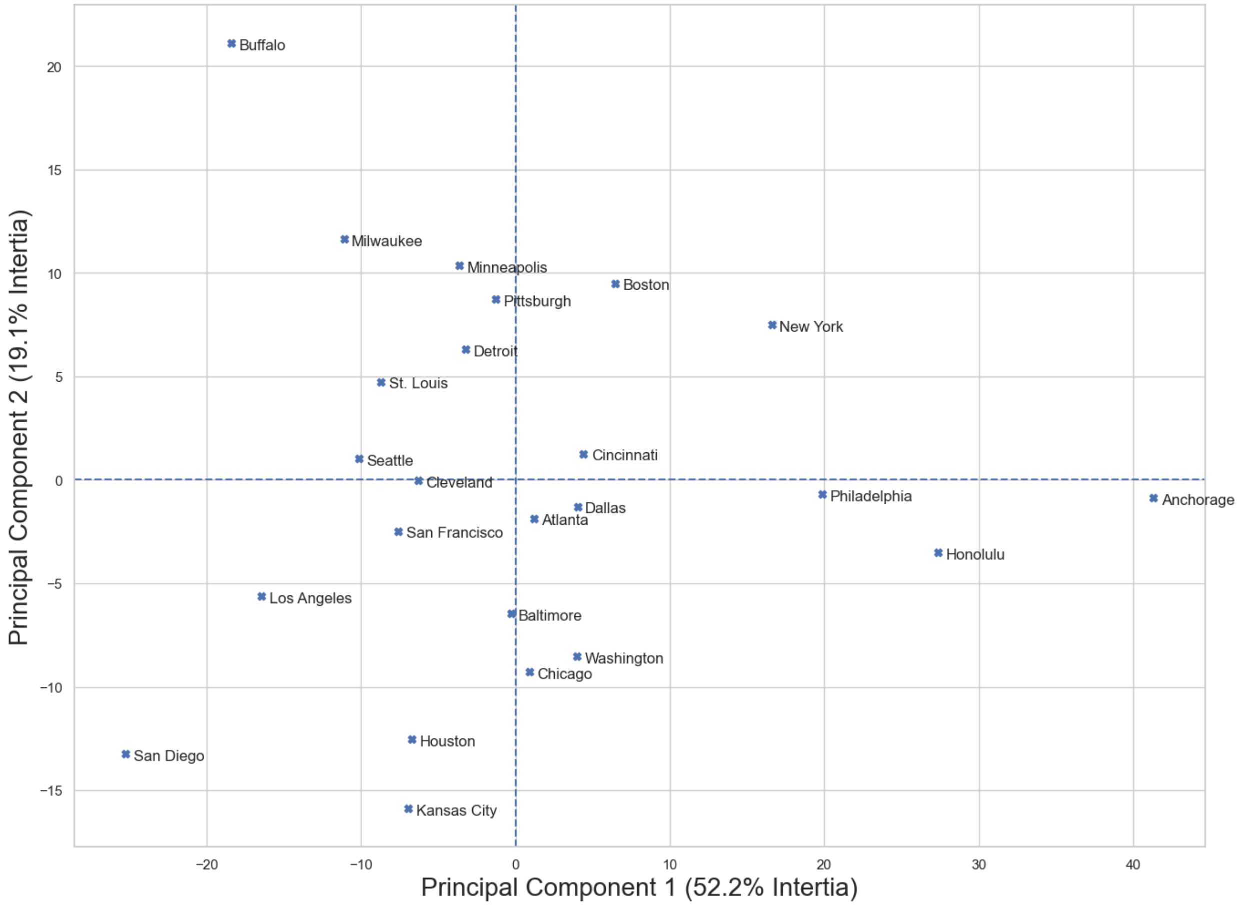
In [316]:
```
type(test1)
```

Out[316]:
```
numpy.ndarray
```

In [406]:
```python
# Preparing dataset
x = np.array(pca_mean_new1_1[0])
y = np.array(pca_mean_new1_1[1])
text = np.array(pca_mean_new1_1.index)
plt.figure(figsize=(16,12))
# plotting scatter plot
plt.scatter(x, y,marker='X',alpha=1)

# Loop for annotation of all points
for i in range(len(x)):
    plt.annotate(text[i], (x[i]+0.5, y[i]-0.3))

#adjusting the scale of the axes
#plt.xlim((-30, 45))
#plt.ylim((-20, 25))
plt.xlabel('Principal Component 1 (52.2% Intertia)',fontsize=20)
plt.ylabel('Principal Component 2 (19.1% Intertia)', fontsize=20)
plt.axvline(0, ls='--')
plt.axhline(0, ls='--')
plt.show()
```



In [ ]:

Standardized Data(不設定n_components)

In [418]:
```python
df[numerical_features].shape
```

Out[418]: (24, 5)

In [421]:
```
scaler = scale(df[numerical_features])
pd.DataFrame(scaler)
#df_std = scaler.fit_transform(df[numerical_features])
#df_std
```

Out[421]:

	0	1	2	3	4
0	3.962979	2.050729	1.194236	1.486313	1.427797
1	-0.249276	-0.065492	0.009641	0.264267	0.847934
2	-1.164984	-0.302579	0.751397	-0.517842	2.035272
3	0.580966	0.619426	-0.244992	-1.299951	0.930771
4	-0.481255	-0.205988	-2.149201	-1.972076	-1.913316
5	-0.163810	-0.416732	0.131422	1.632958	0.613228
6	-0.163810	0.514054	0.596403	-0.750030	0.143815
7	0.007122	-0.399170	-0.167495	-0.175669	-0.905460
8	-0.359161	0.399901	-0.189637	1.303006	0.130009
9	0.287939	-0.302579	-0.455340	-1.471037	0.875546
10	1.521092	1.708270	1.127810	1.620738	0.571809
11	-0.407999	-0.873344	0.673900	0.924172	-0.725979
12	-0.407999	-1.092869	2.002418	-1.116644	-0.256566
13	-0.188229	-1.408985	-0.422127	0.362031	-1.443904
14	-0.627769	-0.276236	-2.326336	0.728645	-0.284179
15	-0.725444	0.391120	-1.008889	-0.456740	-0.090891
16	0.519918	1.629241	0.496764	-0.505621	0.323296
17	0.544337	1.286782	1.061384	0.019858	1.621084
18	-0.188229	0.276967	-0.588192	-0.969999	0.295684
19	-0.188229	-0.214769	-0.466411	-0.615605	-1.485323
20	-0.725444	-2.436362	0.186777	-0.395637	-1.029716
21	0.190264	-0.671381	-0.566050	0.911951	-1.084941
22	-0.762072	-0.601133	-0.820683	0.276488	-0.159922
23	-0.810910	0.391120	1.173201	0.716424	-0.436047

In [478]:
```
pca = prince.PCA(
     n_components=5,
     n_iter=10,
     rescale_with_mean=False,
     rescale_with_std=False,
     copy=True,
     check_input=True,
     engine='sklearn',
     random_state=2
 )
pca2 = pca.fit(df2[numerical_features])
```

In [479]:
```
print(pca2.eigenvalues_)
```

```
[2.43935545 0.92959114 0.83323778 0.53287275 0.26494287]
```

In [480]:
```
#看特徵數量
pca2.eigenvalues_summary
```

Out[480]:

	eigenvalue	% of variance	% of variance (cumulative)
component			
0	2.439	48.79%	48.79%
1	0.930	18.59%	67.38%
2	0.833	16.66%	84.04%
3	0.533	10.66%	94.70%
4	0.265	5.30%	100.00%

In [481]:
```
pca2.column_cosine_similarities_
```

Out[481]:

component	**0**	**1**	**2**	**3**	**4**
variable					
Bread	0.634294	0.002967	0.134465	0.150804	0.077471
Hamburger	0.659784	0.071644	0.138322	0.002723	0.127528
Butter	0.385066	0.009185	0.492075	0.098718	0.014956
Apples	0.206485	0.714573	0.004243	0.073970	0.000729
Tomatoes	0.553727	0.131222	0.064133	0.206658	0.044260

In [482]:
```
pca2.column_correlations
```

Out[482]:

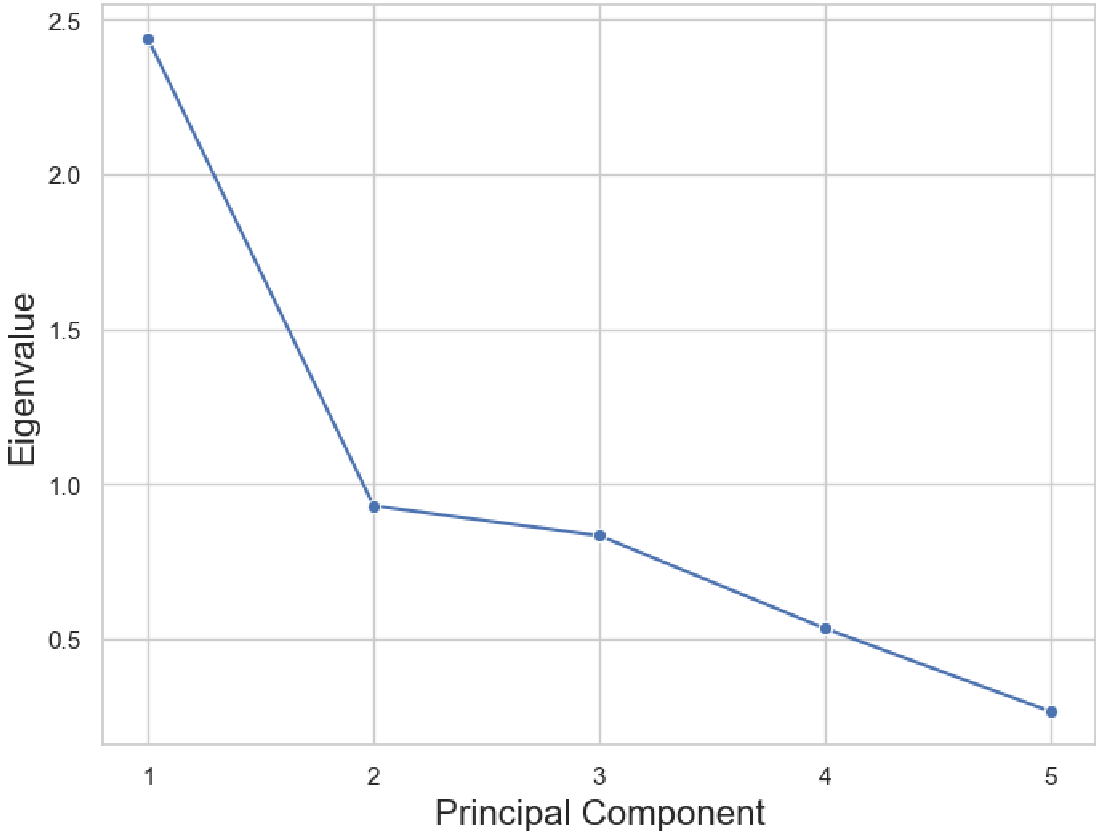
component	**0**	**1**	**2**	**3**	**4**
variable					
Bread	0.796426	-0.054471	-0.366694	-0.388335	-0.278336
Hamburger	0.812271	0.267664	-0.371916	0.052180	0.357110
Butter	0.620537	-0.095838	0.701480	-0.314194	0.122295
Apples	0.454406	-0.845324	-0.065142	0.271975	0.026992
Tomatoes	0.744128	0.362246	0.253246	0.454596	-0.210380

In [483]:
```
pca2.row_coordinates(df2[numerical_features])
```

Out[483]:

component	0	1	2	3	4
0	4.674532	-0.539971	-1.219743	-1.032721	-0.942086
1	0.323536	0.081827	0.350617	0.750290	-0.241064
2	0.366146	1.125827	1.770312	1.349236	-0.260457
3	0.586307	1.652933	-0.323036	-0.064031	-0.391209
4	-2.691778	1.193800	-1.764440	-0.759931	0.285295
5	0.519218	-1.320803	0.390188	0.991082	-0.334328
6	0.271076	0.804306	0.408106	-0.322696	0.488812
7	-0.753021	-0.280745	-0.207609	-0.589556	0.040285
8	0.390528	-0.943408	-0.221303	0.867708	0.441796
9	-0.202286	1.563685	0.005576	0.018349	-0.908808
10	2.856135	-0.929948	-0.397386	-0.212560	0.481889
11	-0.471509	-1.369421	0.770247	-0.243214	0.120009
12	-0.427955	0.403235	2.156495	-1.298756	-0.015528
13	-1.579078	-1.198470	-0.101130	-0.583199	-0.366904
14	-1.311458	-0.555591	-1.553842	1.410007	-0.250544
15	-0.743544	0.616150	-0.635865	0.621345	0.437129
16	1.316738	0.938323	-0.365145	-0.360991	0.808577
17	2.146621	0.812629	0.521023	0.362499	0.189048
18	-0.326972	1.107534	-0.337990	0.195835	-0.017528
19	-1.279764	-0.020950	-0.563453	-0.868814	0.416768
20	-2.168506	-0.693957	1.170177	-0.657286	-0.853537
21	-0.728630	-1.348053	-0.603961	-0.241445	-0.211905
22	-1.023051	-0.344750	-0.143714	0.719092	-0.120095
23	0.256718	-0.754180	0.895877	-0.050243	1.204384

In [484]:
```
#印出陡坡圖
dset = pd.DataFrame()
dset['pca'] = range(1,6)
dset['eigenvalue'] = pd.DataFrame(pca2.eigenvalues_)
plt.figure(figsize=(8,6))
sns.lineplot(x='pca', y='eigenvalue', marker="o", data=dset)
#使用matplotlib模組中plt.xticks()函數設定標線為整數
plt.xticks(np.arange(1, 6, 1))
plt.ylabel('Eigenvalue', fontsize=16)
plt.xlabel('Principal Component', fontsize=16)
plt.show()
```



In [485]:

```
# 繪製解釋變異數圖
plt.plot(range(1, 6), pca2.percentage_of_variance_, 'o-', label='Percent')
plt.plot(range(1, 6), pca2.cumulative_percentage_of_variance_,'o--', color='c', label='Cumulat


plt.xticks(np.arange(1, 6, 1))
plt.ylabel('Percentage')
plt.xlabel('Principal components')
plt.legend(title='Legend:')
plt.axhline(80, color='r', linestyle='--');
```