chambers

February 23, 2024

[1]:
```
from sklearn.linear_model import LinearRegression
import pandas as pd
import numpy as np
np.set_printoptions(precision=2, suppress=True)
```

[2]:
```
def svd(matrix, threshold=0.008):
    U, S, Vt = np.linalg.svd(matrix)
    I = np.where(S > threshold)[0]
    return U[:,I], np.diag(S[I]), Vt[I,:]



# L = np.array([
#     [1.25, 0.83, 0, -0.12],
#     [1.05, 1.13, 0.35, np.nan],
#     [1.12, 1.02, 0.21, np.nan],
#     [1.57, 0.35, -0.56, np.nan],
#     [np.nan, 0.18, 1.02, 0.98]
# ])
# L = L.T
L = np.array([
    [ 1.25,  1.05,  1.12,  1.57,   np.nan],
    [ 0.83,  1.13,  1.02,  0.35,  0.18],
    [ 0.  ,  0.35,  0.21, -0.56,  1.02],
    [-0.12,   np.nan,   np.nan,   np.nan,  0.98]
])
print(L)
```

```
[[ 1.25  1.05  1.12  1.57   nan]
 [ 0.83  1.13  1.02  0.35  0.18]
 [ 0.    0.35  0.21 -0.56  1.02]
 [-0.12   nan   nan   nan  0.98]]
```

[3]:
```
L = L[:-1, :-1]
print(L)
```

```
[[ 1.25  1.05  1.12  1.57]
 [ 0.83  1.13  1.02  0.35]
 [ 0.    0.35  0.21 -0.56]]
```

[4]:
```
r = 1
c = 2
target = L[:,c].copy()
print(target.reshape(-1, 1))
```

```
[[1.12]
 [1.02]
 [0.21]]
```

[5]:
```
def influence(matrix, instances=0, r=1, c=2, party_line=True):
    M = matrix.copy()
    if instances > 0:
        echo = L[:,c].copy()
        chamber = np.tile(echo[:,np.newaxis], (1,instances))
        if not party_line:
            chamber = chamber + np.random.normal(0, 1e-1, chamber.shape)
        M = np.hstack((M, chamber))
    M[r,c] = np.nan
    minor = np.delete(M, c, axis=1)
    return M, minor

M, M_minor = influence(L, instances=0)
print(f"M:\n{M}\nL_m:\n{M_minor}\n")
```

```
M:
[[ 1.25  1.05  1.12  1.57]
 [ 0.83  1.13   nan  0.35]
 [ 0.    0.35  0.21 -0.56]]
L_m:
[[ 1.25  1.05  1.57]
 [ 0.83  1.13  0.35]
 [ 0.    0.35 -0.56]]
```

[6]:
```
def fill(L, L_minor, c=2):
    U, S, Vt = svd(L_minor)
    A = np.dot(U, np.dot(S, Vt))
    k_nan = L[:,c]
    Ln = np.nan_to_num(L, nan=0.0)
    k = Ln[:,c]

    non_nan = ~np.isnan(L[:,c])
    mk = k[non_nan]
    mA = A[non_nan,:]
    d = np.linalg.norm(mA-mk[:,np.newaxis], axis=0)
    nearest_c = np.argmin(d)
    nearest_neighbor = A[:,nearest_c]
    neighbor = nearest_neighbor + (Ln[:,c]-nearest_neighbor) * k / np.linalg.
  ↪norm(k)

    C = mA
    coef, _, _, _ = np.linalg.lstsq(C, mk, rcond=None)
    combo = np.dot(A, coef)
    mixture = combo + (Ln[:,c]-combo) * k / np.linalg.norm(k)
    return L, Ln, L_minor, k_nan, neighbor, mixture, U, S, Vt

M, Mn, M_minor, k_M, neighbor_M, mixture_M, U_M, S_M, Vt_M = fill(M, M_minor,␣
  ↪c=2)

print(f"M:\n{M}\n{Mn}\nM_m:\n{M_minor}\nU:{U_M}\nS:\n{S_M}\nVt:{Vt_M}\n")
print(f"prior:\n{k_M}\ntarget:\n{target}\nEstimates:
  ↪\n{neighbor_M}\n{mixture_M}")
```

```
M:
[[ 1.25  1.05  1.12  1.57]
 [ 0.83  1.13   nan  0.35]
 [ 0.    0.35  0.21 -0.56]]
[[ 1.25  1.05  1.12  1.57]
 [ 0.83  1.13  0.    0.35]
 [ 0.    0.35  0.21 -0.56]]
M_m:
[[ 1.25  1.05  1.57]
 [ 0.83  1.13  0.35]
 [ 0.    0.35 -0.56]]
U:[[-0.86  0.33]
 [-0.51 -0.64]
 [ 0.05 -0.69]]
S:
[[2.61 0.  ]
 [0.   0.93]]
Vt:[[-0.57 -0.56 -0.6 ]
 [-0.12 -0.66  0.74]]

prior:
[1.12  nan 0.21]
target:
[1.12 1.02 0.21]
Estimates:
[1.12 1.13 0.32]
[1.12 1.   0.21]
```

[7]:
```
F, F_minor = influence(L, instances=1)
C, C_minor = influence(L, instances=2, party_line=True)
```

```
F, Fn, F_minor, k_F, neighbor_F, mixture_F, U_F, S_F, Vt_F = fill(F, F_minor,␣
  ↪c=2)
print(f"F:\n{F}\n{Fn}\nF_m:\n{F_minor}\nU:{U_F}\nS:{S_F}\nV^*:\n{Vt_F}\n")
print(f"prior:\n{k_F}\ntarget:\n{target}\nEstimates:
  ↪\n{neighbor_F}\n{mixture_F}")

C, Cn, C_minor, k_C, neighbor_C, mixture_C, _, S_C, _ = fill(C, C_minor, c=2)
print(f"C:\n{C}\n{Cn}\nC_m:\n{C_minor}\nS:{S_C}\n")
print(f"prior:\n{k_C}\ntarget:\n{target}\nEstimates:
  ↪\n{neighbor_C}\n{mixture_C}")
```

```
F:
[[ 1.25  1.05  1.12  1.57  1.12]
 [ 0.83  1.13   nan  0.35  1.02]
 [ 0.    0.35  0.21 -0.56  0.21]]
[[ 1.25  1.05  1.12  1.57  1.12]
 [ 0.83  1.13  0.    0.35  1.02]
 [ 0.    0.35  0.21 -0.56  0.21]]
F_m:
[[ 1.25  1.05  1.57  1.12]
 [ 0.83  1.13  0.35  1.02]
 [ 0.    0.35 -0.56  0.21]]
U:[[ 0.83  0.4 ]
 [ 0.56 -0.6 ]
 [-0.   -0.69]]
S:[[3. 0.]
 [0. 1.]]
V^*:
[[ 0.5   0.5   0.5   0.5 ]
 [ 0.   -0.5   0.81 -0.31]]

prior:
[1.12  nan 0.21]
target:
[1.12 1.02 0.21]
Estimates:
[1.12 1.02 0.21]
[1.12 1.01 0.21]
C:
[[ 1.25  1.05  1.12  1.57  1.12  1.12]
 [ 0.83  1.13   nan  0.35  1.02  1.02]
 [ 0.    0.35  0.21 -0.56  0.21  0.21]]
[[ 1.25  1.05  1.12  1.57  1.12  1.12]
 [ 0.83  1.13  0.    0.35  1.02  1.02]
 [ 0.    0.35  0.21 -0.56  0.21  0.21]]
C_m:
[[ 1.25  1.05  1.57  1.12  1.12]
```

```
 [ 0.83  1.13  0.35  1.02  1.02]
 [ 0.    0.35 -0.56  0.21  0.21]]
S:[[3.36 0.   0.  ]
 [0.   1.03 0.  ]
 [0.   0.   0.01]]

prior:
[1.12  nan 0.21]
target:
[1.12 1.02 0.21]
Estimates:
[1.12 1.02 0.21]
[1.12 1.01 0.21]
```

[8]:
```
instances = 10000
P, P_minor = influence(L, instances=538, party_line=False)
P, Pn, P_minor, k_P, neighbor_P, mixture_P, _, S_P, _ = fill(P, P_minor, c=2)

print(f"P:\n{P}\n{Pn}\nP_m:\n{P_minor}\n")
print(f"S:{S_P}\n")
print(f"prior:\n{k_P}\ntarget:\n{target}\nEstimates:
 ↪\n{neighbor_P}\n{mixture_P}")
```

```
P:
[[1.25 1.05 1.12 … 1.16 1.22 1.17]
 [0.83 1.13  nan … 0.96 1.19 1.09]
 [0.   0.35 0.21 … 0.11 0.17 0.15]]
[[1.25 1.05 1.12 … 1.16 1.22 1.17]
 [0.83 1.13 0.   … 0.96 1.19 1.09]
 [0.   0.35 0.21 … 0.11 0.17 0.15]]
P_m:
[[ 1.25  1.05  1.57 …  1.16  1.22  1.17]
 [ 0.83  1.13  0.35 …  0.96  1.19  1.09]
 [ 0.    0.35 -0.56 …  0.11  0.17  0.15]]

S:[[35.53  0.    0.  ]
 [ 0.    2.61  0.  ]
 [ 0.    0.    2.22]]

prior:
[1.12  nan 0.21]
target:
[1.12 1.02 0.21]
Estimates:
[1.12 0.97 0.21]
[1.12 1.01 0.21]
```

```
[9]: df = pd.read_csv("./vitalstats_ch8_tbl4.csv")
     df["Score"] = pd.to_numeric(df["Score"], errors="coerce")
     df = df.dropna(subset=["Score"])
     latest = df[(df["Year"] == 2016)]
     print(df)
     print(latest.to_string(index=False))
```

```
     Year Chamber          Party  Score
0    1954   House  All Democrats   80.0
1    1955   House  All Democrats   84.0
2    1956   House  All Democrats   80.0
3    1957   House  All Democrats   79.0
4    1958   House  All Democrats   77.0
..    …      …              …     …
373  2012  Senate    Republicans   83.0
374  2013  Senate    Republicans   89.0
375  2014  Senate    Republicans   90.0
376  2015  Senate    Republicans   91.0
377  2016  Senate    Republicans   86.0

[342 rows x 4 columns]
 Year Chamber          Party  Score
 2016   House All Democrats   96.0
 2016   House    Republicans   96.0
 2016  Senate All Democrats   92.0
 2016  Senate    Republicans   86.0
```

```
[10]: print(df[(df["Year"] == 2014)].to_string(index=False))
      # https://www.brookings.edu/articles/vital-statistics-on-congress/
```

```
 Year Chamber          Party  Score
 2014   House All Democrats   94.0
 2014   House    Republicans   95.0
 2014  Senate All Democrats   99.0
 2014  Senate    Republicans   90.0
```