In [17]:
```
import pyreadr
import numpy as np

result = pyreadr.read_r("SMRi ordered - Brazil")   # também lê .RData
obj = next(iter(result.values()))        # pega o único objeto do .rds

# se for um data frame R, vira um pandas.DataFrame
import pandas as pd
df = obj if isinstance(obj, pd.DataFrame) else None
df
```

Out[17]:

	C349_F	C509_F	C349_M	C509_M	C169_F	C61_M	C539_F	C159_F	C169_
0	1.123807	0.661231	0.986634	0.707814	1.221911	0.742698	1.529891	0.683831	1.3146
1	0.443441	0.610168	0.448988	1.158253	0.574836	0.669648	1.016424	0.642775	0.8570
2	0.879448	0.449045	0.890769	0.604989	1.166260	0.811681	0.525437	0.498478	0.8600
3	1.020218	0.422304	1.001927	0.662585	0.601234	1.225795	0.519147	0.668390	0.7399
4	0.443073	0.354335	0.906988	0.692725	0.575117	0.642738	0.642628	0.642295	0.6613
...	...	...	...	...	...	...	...	...	
552	0.795859	0.878455	0.941837	0.501845	0.541944	0.668475	0.851317	0.613098	0.5979
553	0.736944	0.657660	0.301150	0.686377	0.532942	0.702059	0.835282	0.613446	0.6168
554	0.574846	0.596009	0.625308	0.505195	0.645011	0.514871	0.851198	0.707966	0.4493
555	1.210127	0.824201	0.832557	1.017706	0.747300	1.087854	1.233877	1.062437	0.5349
556	0.342973	0.419991	0.913603	1.234261	0.444595	0.794759	0.454757	0.277496	1.0593

557 rows × 30 columns

In [18]:
```
vars_cols = df.var()
vars_cols.head()
```

Out[18]:
```
C349_F    0.674739
C509_F    0.179864
C349_M    0.568566
C509_M    0.175342
C169_F    0.355013
dtype: float64
```

In [19]:
```
df_std = df.copy()

#vamos padronizar as colunas
mu = df_std.mean(axis=0)
sd = df_std.std(axis=0, ddof=0)  # desvio padrão populacional (ddof=0)

# evita divisão por zero (coluna constante). Aqui descartamos constantes.
```

```
keep = sd > 0

# Y é o DataFrame padronizado
Y = ((df_std.loc[:, keep] - mu[keep]) / sd[keep]).fillna(0.0)
print(Y.values, Y.shape)
Y.var().head()
```

```
[[ 0.36497084 -0.18883623  0.27358684 ...  0.17555823  1.919834
  -0.54308607]
 [-0.46404909 -0.30934677 -0.44008088 ... -0.18922218 -0.1473854
  -0.44313775]
 [ 0.06722115 -0.68960224  0.14633648 ...  0.05572086  0.17285073
  -0.75892984]
 ...
 [-0.30393368 -0.34276097 -0.20603454 ... -1.02813628 -0.87380289
  -0.46704425]
 [ 0.47015061  0.19577766  0.06906562 ... -0.29516127  0.58097297
  -0.02770818]
 [-0.58646782 -0.75816981  0.17664611 ... -0.04401604  1.72514341
  -0.75674525]] (557, 30)
```

Out[19]:
```
C349_F    1.001799
C509_F    1.001799
C349_M    1.001799
C509_M    1.001799
C169_F    1.001799
dtype: float64
```

In [20]:
```
# Agora sim, podemos fazer a SVD
from numpy.linalg import svd

# X = M S Vt
M, S, Vt = svd(Y.values, full_matrices=False)
print(M.shape, S.shape, Vt.shape)  # M:(n×q), S:(q,), Vt:(q×q)
```

```
(557, 30) (30,) (30, 30)
```

In [21]:
```
# Gavish-Donoho para escolher K
n, q = Y.shape

beta = min(q / n, n / q)
sigma_median = np.median(S)
omega = 0.56*beta**3 - 0.95*beta**2 + 1.82*beta + 1.43
tau = omega * sigma_median
K = np.sum(S >= tau)
print(f"n={n}, q={q}, beta={beta:.3f}, omega={omega:.3f}, tau={tau:.3f}, K={K}")
```

```
n=557, q=30, beta=0.054, omega=1.525, tau=20.351, K=6
```