Generalized Procrustes Analysis

The following notes on generalized Procrustes Analysis are based on: I. Borg and Patrick J. F. Groenen, Modern Multidimensional Scaling (2nd ed.) (Springer, NY, 2005). Chapter 21.

While scaling and translation are complicated (indeed, they are complicated for any non-orthogonal transformation), in `Procrustes` we've elected to use a simple scaling/translation methodology. So we will assume that the input matrices, $\mathbf{A}_k$ are scaled and translated by the normal rules, if that is desired. The following method corresponds to the *first* approach mentioned in the book of Borg and Groenen.

The objective function is obtained by transforming multiple matrices in generalized Procrustes analysis. So:

$$\underbrace{\min}_{\mathbf{T}_1,\mathbf{T}_2,\ldots\mathbf{T}_K} \sum_{1\le k<l\le N} \|\mathbf{A}_k\mathbf{T}_k - \mathbf{A}_l\mathbf{T}_l\|^2 = \underbrace{\min}_{\mathbf{T}_1,\mathbf{T}_2,\ldots\mathbf{T}_K} \sum_{1\le k<l\le N} \mathrm{Tr}\left[(\mathbf{A}_k\mathbf{T}_k - \mathbf{A}_l\mathbf{T}_l)^\dagger (\mathbf{A}_k\mathbf{T}_k - \mathbf{A}_l\mathbf{T}_l)\right]$$

can be solved by rewriting this as

$$\underbrace{\min}_{\mathbf{T}_1,\mathbf{T}_2,\ldots\mathbf{T}_K} \frac{1}{2} \left\| \sum_{1\le k\le K} \left(\mathbf{A}_k\mathbf{T}_k - \sum_{\substack{1\le l\le N \\ l\ne k}} \mathbf{A}_l\mathbf{T}_l \right) \right\|^2$$

Then you can decide to optimize this expression either by reference to the mean of the matrices or by optimizing one transformation at a time. The former is arguably easier to implement (and generalize), though it isn't what we are doing right now. In that case, for each $k = 1, 2, \ldots, K$, one solves the single-matrix Procrustes problem,

$$\underbrace{\min}_{\mathbf{T}_k^{(i)}} \left\| \left(\mathbf{A}_k\mathbf{T}_k^{(i)} - \sum_{\substack{1\le l\le N \\ l\ne k}} \mathbf{A}_l\mathbf{T}_l^{(i-1)} \right) \right\|^2$$

where it is sensible (but not required) to initialize the transformations to the identity, $\mathbf{T}_k^{(0)} = \mathbf{I}$. As long as the optimization is linear, this algorithm converges, and is equivalent to the usual approach. This notation is a little bad, because in actually, for $l < k$, you would be using the latest transformation, $\mathbf{T}_l^{(i)}$ in the summation.

The nice thing about this algorithm is that it generalizes to *any* generalized Procrustes problem. If you pass multple matrices to any Procrustes method, then you end up with the generic problem

$$\underbrace{\min}_{\substack{\mathbf{T}_1,\mathbf{T}_2,\ldots\mathbf{T}_K \\ \mathbf{S}_1,\mathbf{S}_2,\ldots\mathbf{S}_K}} \sum_{1\le k<l\le N} \left\|\mathbf{S}_k\mathbf{A}_k\mathbf{T}_k - \mathbf{S}_l\mathbf{A}_l\mathbf{T}_l\right\|^2$$

which can be (approximately) solved by iteratively performing, for $k = 1, 2, \ldots, K$,

$$\underbrace{\min}_{\mathbf{T}_k^{(i)}} \left\| \left(\mathbf{S}_k^{(i-1)}\mathbf{A}_k\mathbf{T}_k^{(i)} - \sum_{\substack{1\le l\le N \\ l\ne k}} \mathbf{S}_l^{(i-1)}\mathbf{A}_l\mathbf{T}_l^{(i-1)} \right) \right\|^2$$

then

$$\underbrace{\min}_{\left(\mathbf{S}_k^{(i)}\right)^\dagger} \left\| \left(\left(\mathbf{T}_k^{(i)}\right)^\dagger \mathbf{A}_k^\dagger \left(\mathbf{S}_k^{(i)}\right)^\dagger - \sum_{\substack{1\le l\le N \\ l\ne k}} \left(\mathbf{T}_l^{(i-1)}\right)^\dagger \mathbf{A}_l^\dagger \left(\mathbf{S}_l^{(i-1)}\right)^\dagger \right) \right\|^2$$

until it converges. This notation is a little bad, because in actually, for $l < k$, you would be using the latest transformations, $\mathbf{T}_l^{(i)}$ and $\mathbf{S}_l^{(i)}$ in the summation. But breaking the sum into two parts just to make this explicit seems undesirable.

Proposal

Problem Description

$$\underbrace{\min}_{\substack{\mathbf{T}_1,\mathbf{T}_2,\ldots\mathbf{T}_K \\ \mathbf{S}_1,\mathbf{S}_2,\ldots\mathbf{S}_K}} \sum_{1\le k<l\le N} \left\|\mathbf{S}_k\mathbf{A}_k\mathbf{T}_k - \mathbf{S}_l\mathbf{A}_l\mathbf{T}_l\right\|^2$$

> Note: the traditional problem corresponds to $N_A = 2$ with $\mathbf{S}_2 = \mathbf{T}_2 = \mathbf{I}$.

Input

1. `A` A list of matrices, $\{\mathbf{A}\}_{k=1}^{N_A}$. These are matrices to be transformed.
2. `S_type` A list of length N_A specifying the *types* of left-hand-side transformations for each $\mathbf{A}_k$ matrix. If only one element (not a list) is passed, ideally we could just assume every element in the list was equal to that element. The types we currently support are:
   - identity (no transformation; default)
   - orthogonal
   - rotational
   - symmetric
   - permutation

3. `T_type` A list of length N_A specifying the *types* of right-hand-side transformations. If only one element (not a list) is passed, ideally we could just assume that every element in the list was equal to that element. The types we currently support are:
4. `S_eq_T` A list of *logical* variables indicating whether the left-hand-side and right-hand-side transformations are the same, $\mathbf{S}_k = \mathbf{T}_k$. Optional argument; default to `False`.
5. Optional argument. Lists of flags to indicate options for translation, scaling, zero-row-removal, zero-column-removal, and zero-padding. Default as usual.
6. `S` and `T` Initial guesses for the $\mathbf{S}_k$ and $\mathbf{T}_k$. Default to identity. We do not help users set this up; if users want to do some fancy-schmancy permutation Procrustes guessing, they are responsible for doing that themselves, potentially using utilities we have already provided. We may provide utilities (later) to help with this but it is not part of this issue/functionality.

Check

1. If orthogonal or 2-sided permutation with the same transformation, the matrices must all have the same shape and be square (and symmetric in the two-sided-orthogonal-with-one-transformation case); otherwise all matrices must have the same shape. The default options for zero-padding should ensure this.
2. Input transformation matrices should be checked to ensure that they are of the right type; if Procrustes is used to find the closest matrix of the correct type. If $\mathbf{S}_k = \mathbf{T}_k$ is expected but not satisfied, both inputs are replaced by their average, then the closest transformation matrix of the desired type is constructed. Some user output/warnings should indicate what types of work had to be done to initialize the problem appropriately.
3. At least one element of `S_type` and `T_type` should be non-identity, or otherwise there is nothing to do.
4. All of the other Procrustes methods can be tested against since they are special cases of this one with $N_A = 2$.
5. If `S_eq_T(k) = True`, then since the two transformations are supposed to be the same, it must be that `S_type(k) = T_type(k)`. Otherwise print an informative error message and fail.

Algorithm (Generalized Flip-Flop Algorithm)

This algorithm is a greedy-ish fixed-point iteration algorithm. When the problem is convex, the solution is found. Otherwise a local minima is found.

1. For each $\mathbf{A}_k$ matrix, you should shift, scale, and add/remove zero rows as instructed.
2. For each $k = 1, 2, \ldots, N_A$ with $\mathbf{T}_{\text{type},k}$ not equal to the Identity matrix.Use the one-sided Procrustes method of type $\mathbf{T}_{\text{type},i}$ to solve the problem:

$$\underbrace{\min}_{\mathbf{T}_k^{(i)}} \left\| \left(\mathbf{S}_k^{(i-1)} \mathbf{A}_k \mathbf{T}_k^{(i)} - \sum_{\substack{1 \le l \le N \\ l \neq k}} \mathbf{S}_l^{(i-1)} \mathbf{A}_l \mathbf{T}_l^{(i-1)} \right) \right\|^2$$

or if `S_eq_T(i) = True`, then use the two-sided Procrustes method of type $\mathbf{T}_{\text{type},k}$ to solve the problem:

$$\underbrace{\min}_{\mathbf{T}_k^{(i)}} \left\| \left(\left(\mathbf{T}_k^{(i)}\right)^{\dagger} \mathbf{A}_k \mathbf{T}_k^{(i)} - \sum_{\substack{1 \le l \le N \\ l \neq k}} \mathbf{S}_l^{(i-1)} \mathbf{A}_l \mathbf{T}_l^{(i-1)} \right) \right\|^2$$

3. For each $k = 1, 2, \ldots, N_A$ with $\mathbf{S}_{\text{type},k}$ not equal to the Identity matrix. Use the one-sided Procrustes method of type $\mathbf{S}_{\text{type},k}$ to solve the problem:

$$\underbrace{\min}_{\left(\mathbf{S}_k^{(i)}\right)^{\dagger}} \left\| \left(\left(\mathbf{T}_k^{(i)}\right)^{\dagger} \mathbf{A}_k^{\dagger} \left(\mathbf{S}_k^{(i)}\right)^{\dagger} - \sum_{\substack{1 \le l \le N \\ l \neq k}} \left(\mathbf{T}_l^{(i-1)}\right)^{\dagger} \mathbf{A}_l^{\dagger} \left(\mathbf{S}_l^{(i-1)}\right)^{\dagger} \right) \right\|^2$$

- or if `S_eq_T(i) = True`, then use the two-sided Procrustes method of type $\mathbf{T}_{\text{type},k}$ to solve the problem:

$$\underbrace{\min}_{\mathbf{T}_k^{(i)}} \left\| \left(\left(\mathbf{T}_k^{(i)}\right)^{\dagger} \mathbf{A}_k \mathbf{T}_k^{(i)} - \sum_{\substack{1 \le l \le N \\ l \neq k}} \mathbf{S}_l^{(i-1)} \mathbf{A}_l \mathbf{T}_l^{(i-1)} \right) \right\|^2$$

4. If not converged, go back to step 2.