Derivative-Free Gradient Projection Factor Rotation

The `GPArotateDF` Package

Author: Coen A. Bernaards

Principle of derivative-Free Factor Rotation

The gradient projection algorithm package *GPArotation* consists of wrapper functions and functions that compute the gradients, G_q, needed for minimization. For all functions included in the *GPArotation* package, the gradients are included in the `vgQ` routines. For example, the `vgQ.quartimax` function provides the gradients G_q for quartimax rotation. Examples of gradient derivations, computations are provided in Jennrich (2001) and Jennrich (2002), as well as Bernaards & Jennrich (2005). However, the derivation of the gradient can be quite involved for complex rotation criteria. In such cases, a derivative free version of gradient projection algorithm can be used for minimization of the criterion function, *without* the need for a gradient. Details of the methods are described in Jennrich (2004). The method is implemented in the package *GPArotateDF* that may be downloaded and installed.

To perform derivative-free rotation, the main algorithms `GPForth.df` and `GPFoblq.df` are available for orthogonal and oblique rotation, respectively. Both are minimization algorithms. The algorithms differ from the regular algorithms by the inclusion of the numerical derivates G_f for the rotation criteria in `GPForth.df` and `GPFoblq.df`. The algorithms require: an initial loadings matrix `A` and a rotation method. Optional are initial rotation matrix `Tmat` (default is the identity matrix). Other arguments needed for individual rotations are applied in the same way as in the *GPArotation* package.

The rotation method is provided between quotation marks, and refers to the name of the ff-function. For example, the `method = "varimax"` through `GPForth.df` calls the `ff.varimax` function. The ff-functions are the derivative-free analogues of the GPArotation vgQ functions. The output of `ff.varimax` is the rotation criteria value, `f`, and the `Method` name, e.g. `DF-Varimax`.

New rotation functions need to be programmed as `ff.newmethod`. The only required input is an initial loadings matrix `A`, and any potential additional arguments. The output consist of the value `f` of the criterion, and the `Method` name (the `GPForth.df` and `GPFoblq.df` algorithms expect this included in the result).

Derivative-free quartimax rotation

As an example, consider quartimax rotation. Gradient projection quartimax orthogonal rotation seeks to minimize the sum of all loadings raised to power 4. Thus, using the notation of Bernaards & Jennrich (2005) (page 682), the criterion f for minimization is calculated as

$$f = Q(\Lambda) = -\frac{1}{4}\sum_i \sum_r \lambda_{ir}^4.$$

Derivative-free quartimax rotation using `ff.quartimax` is then very simple

```
> library(GPArotateDF)
> ff.quartimax<- function(L){
```

```
    f = -sum(L^4) / 4
    list(f = f, Method = "DF-Quartimax")
  }
> data(Harman, package="GPArotation")
> GPForth.df(Harman8, method="quartimax")
```

Of course, for quartimax, the gradient is easy to derive and regular rotation is a better choice.

Rotation when the derivative is complicated: cubimax

Sometimes the gradient is hard to derive. For example, a criterion that seeks to minimize loadings to the power 3, the absolute value is needed for a meaningful result.

$$f = Q(\Lambda) = -\sum_i \sum_r |\lambda_{ir}^3|.$$

While the gradient may be complicated, the derivative-free function for minimization is straightforward.

```
> ff.cubimax<- function(L){
    f = -sum(abs(L^3))
    list(f = f, Method = "DF-Cubimax")
  }
> GPForth.df(Harman8, method="cubimax")
```

Results differ from quartimax and varimax rotation

Rotation when an algorithm is involved: Forced Simple Structure

in certain cases the derivate is so poorly defined that deriving the `vgQ` function is a non-starter. For example, an algorithm that updates a weight matrix that, when multiplied with the loadings matrix provides a rotation criterion to be minimized.

The algorithm Forced Simple Structure chooses a weight matrix focused on the lowest loadings. The rotation criterion value f is minimized representing a rotated factor pattern which many low loadings, restricted to each factor having at least some salient loadings. In each iteration, the weight matrix `Imat` gets weight 1 at the lowest factor loadings, and 0 elsewhere.

Assume we have `p` items, and `m` factors (for a `p x m` loadings matrix). In each iteration, first the lowest loadings get weight 1. Next, for each pair `(i,j)` of factors, lowest loadings get weight 1 until there are at least `(m + kij)` items with weight 1 on a single factor `i` or `j` (but not the other factor), or not enough loadings are left to get weight 1. Possible values for `kij = (0, ..., [p - m] )` and defaults to 2. Forced Simple Structure is most effective when `kij` has a lower value. For each increase of 1, an additional `(m)` loadings get weight 1. The criterion `f` minimizes the squared loadings for low loadings ("non-salient"). Salient loadings are therefor increased as the sum of squared non-salient loadings is minimized.

```
> ff.fss <- function(L, kij=2){
    m <- ncol(L)
    p <- nrow(L)
```

```
    zm <- m + kij
    Imat <- matrix(0, p, m)
    for (j in 1:m){
      Imat[abs(L[,j]) <= sort(abs(L[,j]))[zm],j] <- 1 }
    for (i in 1:(m-1)){
      for (j in (i+1):m){
        nz <- sum( (Imat[,i] + Imat[,j]) ==1)
        while (nz < zm && sum(Imat[ ,c(i,j)]) < m * 2){
              tbc <- c(abs(L[,i]), abs(L[,j]))
              tbcs <- sort(tbc [c(Imat[,i], Imat[,j])==0])[1]
              Imat[abs(L) == tbcs] <- 1
              nz <- sum( (Imat[,i] + Imat[,j]) ==1)
        }
      }
    }
    Method <- paste("DF-Forced Simple Structure (kij = ",kij,")", sep="")
    f <- sum(Imat*L^2)
    list(f = f, Imat = Imat,
         Method = Method)
  }
> data(WansbeekMeijer, package = "GPArotation")
> z <- factanal(covmat = NetherlandsTV, factors = 3, rotation = "none")
> fssT.df(loadings(z), kij = 3)
> # which loadings get weight 1 in the first iteration?
> ff.fss(loadings(z), kij = 3)$Imat
```

The added `sum(Imat) < m * 2` requirement was added to avoid infinite looping. It is useful to consider random starts as the rotation tends to have many local minima. The method works both orthogonal and oblique.

Examples of other ff-functions

Writing ff-functions is straightforward because only the criterion value is needed. Here are a few additional examples of ff-functions. For all vgQ-functions exist and are preferable to be used.

The oblique rotation criterion for oblimax

```
> ff.oblimax <- function(L){
    f <- -(log(sum(L^4))-2*log(sum(L^2)))
    list(f = f,
         Method = "DF-Oblimax")
  }
```

Entropy criterion for orthogonal rotation

```
> ff.entropy <- function(L){
    f <- -sum(L^2 * log(L^2 + (L^2==0)))/2
    list(f = f,
         Method = "DF-Entropy")
  }
```

Simplimax that works well in oblique rotation

```
> ff.simplimax <- function(L,k=nrow(L)){
    # k: Number of close to zero loadings
    Imat <- sign(L^2 <= sort(L^2)[k])
    f <- sum(Imat*L^2)
    list(f = f,
         Method = "DF-Simplimax")
  }
```

Target rotation. Requires both a weight matrix and a target matrix. Target rotation can be both orthogonal and oblique.

```
> ff.pst <- function(L,W,Target){
    # Needs weight matrix W with 1's at specified values, 0 otherwise
    # e.g. W = matrix(c(rep(1,4),rep(0,8),rep(1,4)),8).
    # When W has only 1's this is procrustes rotation
    # Needs a Target matrix Target with hypothesized factor loadings.
    # e.g. Target = matrix(0,8,2)
    Btilde <- W * Target
    f <- sum((W*L-Btilde)^2)
    list(f = f,
          Method = "DF-PST")
  }
```

References

Bernaards, C. A., & Jennrich, R. I. (2005). Gradient Projection Algorithms and Software for Arbitrary Rotation Criteria in Factor Analysis. *Educational and Psychological Measurement*, 65(5), 676–696. doi: 10.1177/0013164404272507

Jennrich, R. I. (2002). A simple general procedure for orthogonal rotation. *Psychometrika*, 66(3), 289–306. doi: 10.1007/BF02294840

Jennrich, R. I. (2002). A simple general method for oblique rotation. *Psychometrika*, 67(3), 7–19. doi: 10.1007/BF02294706

Jennrich, R. I. (2004). Derivative free gradient projection algorithms for rotation. *Psychometrika*, 69(3), 475–480. doi: 10.1007/BF02295647