

Gradient Projection Factor Rotation

The GPArotation Package

Author: Coen A. Bernaards

The **GPArotation** package provides gradient projection algorithms for orthogonal and oblique rotation of factor loading matrices in exploratory factor analysis. It implements a comprehensive set of rotation criteria and supports multiple random starts to avoid local minima. This vignette introduces the main functionality of the package, with examples of common use cases. For a complete list of available rotation criteria and their references, see the Rotation Criteria Reference section at the back of this document.

In R, the functions in this package are made available with

```
> library("GPArotation")
```

The most complete reference for the software is Bernaards & Jennrich (2005). The original 2005 implementations in four platforms are preserved for historical reference:

MATLAB code, Splus code, SAS code, and SPSS code. These implementations predate the R package and reflect the algorithm as originally published. The R package has been updated since then; see the **NEWS** file for a full history of changes.

A clear and accessible introduction to gradient projection algorithms for factor rotation is provided in Mansolf & Reise (2016).

Basic Usage

Getting Started

Factor rotation aims to simplify interpretation by rotating the loadings matrix. In practice, most rotations are done by minimizing a criterion function. This package provides algorithms that can minimize any rotation criteria as long as a gradient is available. The loading matrix is typically obtained from a factor analysis routine such as **factanal**. A rotation is performed by calling the rotation function directly, or by calling one of the wrapper functions **GPFRSorth** or **GPFRSoblq** for orthogonal and oblique rotation, respectively.

Under the hood, rotations are computed using the Gradient Projection Algorithm, implemented in **GPForth** for orthogonal rotation and **GPFoblq** for oblique rotation. These functions were updated in version 2026.4-1 with performance improvements and improved code clarity; results are numerically identical to prior versions.

```
> data("Harman", package = "GPArotation")
> # Calling a rotation directly
> qHarman <- quartimax(Harman8)
> # Equivalently, via the wrapper function
> qHarman <- GPFRSorth(Harman8, method = "quartimax")
> # Two equivalent ways to access the rotated loadings
> loadings(qHarman)           # via extractor function (recommended)
> qHarman$loadings            # via direct list access
```

The rotated loadings matrix is the pattern matrix. The extractor function **loadings()** is preferred over direct list access for forward compatibility.

Recovery of the Unrotated Loadings Matrix

The rotated loadings matrix L and the rotation matrix T are related to the unrotated matrix A by:

- Orthogonal rotation: $L = AT$, so $A = LT'$ (since T is orthogonal, $T' = T^{-1}$)
- Oblique rotation: $L = A(T')^{-1}$, so $A = LT'$

In both cases $A = LT'$, where T' denotes the transpose of the rotation matrix. For orthogonal rotation this simplifies further since $T'T = I$.

These relationships are useful for verifying results, switching rotation criteria without re-extracting factors, and reproducing published analyses. The definitions follow Bernaards & Jennrich (2005) (page 678).

```

> data("CCAI", package = "GPArotation")
> fa.un <- factanal(factors = 3, covmat = CCAI_R, n.obs = 461,
                    rotation = "none")
> res.quart <- quartimax(fa.un)
> max(abs(res.quart$loadings %*% t(res.quart$Th) - fa.un$loadings))

[1] 5.551115e-16

> res.obli <- oblimin(fa.un, normalize = TRUE, randomStarts = 15)
> max(abs(res.obli$loadings %*% t(res.obli$Th) - fa.un$loadings))

[1] 2.220446e-16

> max(abs(res.obli$loadings - fa.un$loadings %*% solve(t(res.obli$Th))))

[1] 3.330669e-16

```

All three expressions should be zero or at machine epsilon, confirming that the rotated solution is consistent with the unrotated extraction. The factor correlation matrix for oblique rotation is Bornaards & Jennrich (2005) (page 695):

```

> max(abs(res.obli$Phi - t(res.obli$Th) %*% res.obli$Th))

[1] 0

```

This should also be zero at machine epsilon. Note that `attr(res.obli, "A_unrotated")` stores the unrotated matrix directly in the `GPArotation` object, so manual recovery is rarely needed in practice.

Output and Display

The raw output from `GPArotation` functions returns loadings in the order determined by the algorithm, which depends on the starting rotation matrix. This means that with different random starts, the same solution may appear with factors in a different order or with reversed signs. The `print` method sorts factors by descending variance explained and adjusts signs so that the dominant loadings are positive, making visual inspection and comparison easier. Importantly, this is a display transformation only — the underlying object is not modified.

```

> set.seed(334)
> res <- quartimin(Harman8, normalize = TRUE, randomStarts = 100)
> # Raw unsorted loadings
> loadings(res)

```

Loadings:

	CF1	CF2
height		-0.892
arm.span		-0.954
forearm		-0.929
lower.leg		-0.877
weight	-0.925	
bitro.diameter	-0.821	
chest.girth	-0.765	
chest.width	-0.683	

	CF1	CF2
SS loadings	2.604	3.362

```

> # Sorted loadings via print (factors reordered and signs adjusted)
> res.sorted <- print(res)

```

Oblique rotation method Quartimin converged at lowest minimum.
 Of 100 random starts 100% converged, 100% at the same lowest minimum.
 Loadings at lowest minimum:

	CF1	CF2
height	0.892	
arm.span	0.954	
forearm	0.929	
lower.leg	0.877	
weight		0.925
bitro.diameter		0.821
chest.girth		0.765
chest.width		0.683

	CF1	CF2
SS loadings	3.364	2.604
AUC	0.821	0.842
FSI	0.142	0.155

Phi:

	CF1	CF2
CF1	1.000	0.473
CF2	0.473	1.000

```
> # Once sorted, repeated calls to print are stable
> max(abs(print(res.sorted)$loadings - res.sorted$loadings)) == 0 # TRUE
```

Oblique rotation method Quartimin converged at lowest minimum.
 Of 100 random starts 100% converged, 100% at the same lowest minimum.
 Loadings at lowest minimum:

	CF1	CF2
height	0.892	
arm.span	0.954	
forearm	0.929	
lower.leg	0.877	
weight		0.925
bitro.diameter		0.821
chest.girth		0.765
chest.width		0.683

	CF1	CF2
SS loadings	3.364	2.604
AUC	0.821	0.842
FSI	0.142	0.155

Phi:

	CF1	CF2
CF1	1.000	0.473
CF2	0.473	1.000

[1] TRUE

Pattern and Structure Matrices

Two Interpretations of the Loading Matrix

The loading matrix Λ has two complementary interpretations that correspond directly to the pattern and structure matrices in oblique rotation.

First interpretation: regression. The conditional expectation $E(X|f) = \mu + \Lambda f$ is the linear regression of X on f . The loading λ_{ij} is the expected change in item i when factor j is increased by one unit, holding other factors constant. This is the *pattern matrix* — the regression coefficients of items on factors.

Second interpretation: correlation. Under the factor model assumptions, $Cov(X, f) = \Lambda\Phi$. If X is standardized, $Corr(X, f) = \Lambda\Phi$. This is the *structure matrix* — the correlations between items and factors. Correlations tend to be consistent across studies.

When factors are orthogonal ($\Phi = I$) the two interpretations coincide: $\Lambda\Phi = \Lambda$. When factors are oblique the structure matrix $\Lambda\Phi$ inflates the apparent relationships between items and factors through the factor intercorrelations, while the pattern matrix Λ shows the unique contribution of each factor net of those intercorrelations.

Using GPArotation to obtain them

The `summary` method returns the raw unsorted loadings exactly as produced by the algorithm. For oblique rotations, it also prints the structure matrix (loadings $\times \Phi$) when `Structure = TRUE` (the default). This is useful for comparing results across software or reproducing published values.

```
> res.obli <- oblimin(Harman8, normalize = TRUE, randomStarts = 100)
> # Pattern matrix (unsorted)
> summary(res.obli, Structure = FALSE)
```

Oblique rotation method Oblimin Quartimin converged in 11 iterations.
 Algorithm: bb | fwindow: 10 | Iterations:
 Of 100 random starts 100% converged, 100% at the same lowest minimum.

Loadings:

	CF1	CF2
height	0.892	
arm.span	0.954	
forearm	0.929	
lower.leg	0.877	
weight		0.925
bitro.diameter		0.821
chest.girth		0.765
chest.width		0.683

	CF1	CF2
SS loadings	3.364	2.604
AUC	0.821	0.842
FSI	0.142	0.155

Hyperplane total: 8 of 8 (100 %) at cutoff 0.1

Post-Hoc Simplicity Suite (overall solution):

Hoffman Index: 0.652
 Gini Coefficient: 0.479
 Bentler Index: 0.435

Phi:

	CF1	CF2
CF1	1.000	0.473
CF2	0.473	1.000

```
> # Structure matrix
> summary(res.obli, Structure = TRUE)
```

Oblique rotation method Oblimin Quartimin converged in 11 iterations.
 Algorithm: bb | fwindow: 10 | Iterations:
 Of 100 random starts 100% converged, 100% at the same lowest minimum.
 Pattern (loadings) - see Structure (correlations) below:

Loadings:

	CF1	CF2
--	-----	-----

height	0.892
arm.span	0.954
forearm	0.929
lower.leg	0.877
weight	0.925
bitro.diameter	0.821
chest.girth	0.765
chest.width	0.683

	CF1	CF2
SS loadings	3.364	2.604
AUC	0.821	0.842
FSI	0.142	0.155

Correlations:

	CF1	CF2
height	0.918	0.478
arm.span	0.943	0.428
forearm	0.907	0.393
lower.leg	0.893	0.448
weight	0.451	0.931
bitro.diameter	0.371	0.813
chest.girth	0.309	0.740
chest.width	0.409	0.724

CF1	CF2
-----	-----

Hyperplane total: 8 of 8 (100 %) at cutoff 0.1

Post-Hoc Simplicity Suite (overall solution):

Hoffman Index:	0.652
Gini Coefficient:	0.479
Bentler Index:	0.435

Phi:

	CF1	CF2
CF1	1.000	0.473
CF2	0.473	1.000

To illustrate the distinction concretely, consider item 1 (row 1) from the oblimin rotation of the Harman 8-variable physical measurements dataset. In the pattern matrix, the loadings are 0.892 on Factor 1 and 0.056 on Factor 2. The pattern coefficient 0.892 is the regression coefficient of item 1 on Factor 1, controlling for Factor 2 — it represents the unique contribution of Factor 1 to item 1, net of the shared variance between the two factors. The near-zero cross-loading of 0.056 indicates that item 1 is primarily associated with Factor 1 after accounting for factor intercorrelations.

In the structure matrix, the same item has loadings of 0.918 on Factor 1 and 0.478 on Factor 2. The structure coefficient 0.478 is the simple correlation between item 1 and Factor 2 — substantially larger than the pattern cross-loading of 0.056. This inflation occurs because Factor 1 and Factor 2 are correlated, and that correlation carries over into the structure coefficients. An analyst relying solely on the structure matrix might incorrectly conclude that item 1 has a meaningful relationship with Factor 2.

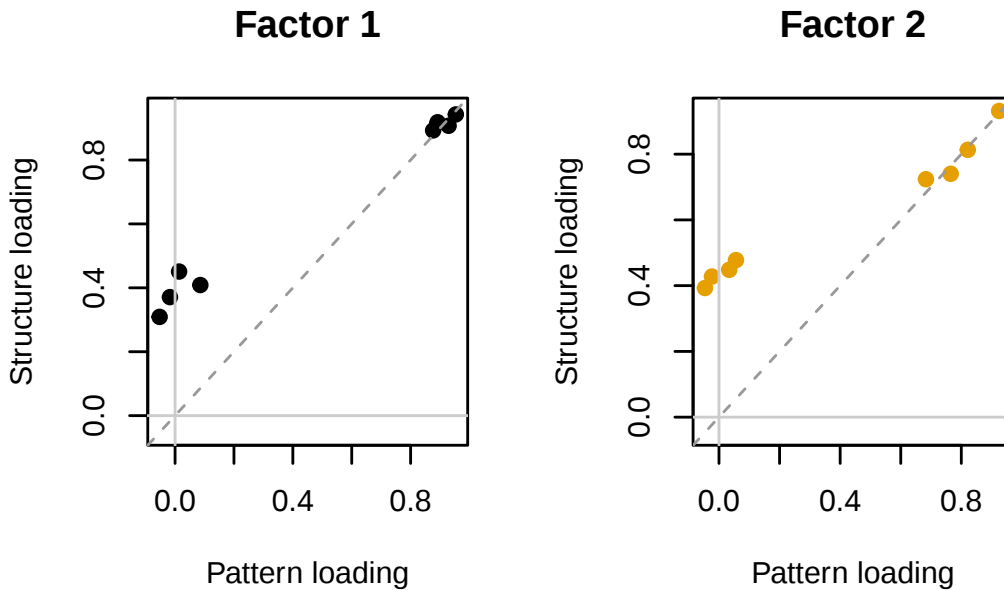
In this solution the factor intercorrelation is $\phi = 0.473$. This moderate correlation between the two factors is sufficient to inflate the structure coefficients noticeably — the cross-loading of item 1 on Factor 2 increases from 0.056 in the pattern matrix to 0.478 in the structure matrix, a difference of 0.422 attributable entirely to the factor intercorrelation.

This is why the pattern matrix is generally preferred for interpretation in oblique rotation: it shows the unique contribution of each factor to each item, net of the shared variance among factors. The structure matrix is a useful diagnostic check — large discrepancies between pattern and structure coefficients signal that factor intercorrelations are substantially inflating apparent relationships.

The relationship between pattern and structure coefficients can be visualized by plotting one against the other for

each factor. Points on the identity line (dashed) have equal pattern and structure coefficients — no inflation from factor intercorrelations. Points above the line have structure coefficients inflated by the factor intercorrelations.

```
> plotPatternStructure <- function(pattern, structure,
                                   labels = NULL,
                                   main = "Pattern vs Structure") {
  k <- ncol(pattern)
  col <- palette.colors(k, palette = "Okabe-Ito")
  par(mfrow = c(1, k))
  for (j in 1:k) {
    lims <- range(c(pattern[, j], structure[, j]))
    plot(pattern[, j], structure[, j],
         xlim = lims, ylim = lims,
         xlab = "Pattern loading",
         ylab = "Structure loading",
         main = paste(if (!is.null(labels)) labels[j]
                       else paste("Factor", j)),
         pch = 19, col = col[j])
    abline(0, 1, lty = 2, col = "grey60")
    abline(h = 0, col = "grey80")
    abline(v = 0, col = "grey80")
  }
}
```



With a factor intercorrelation of $\phi = 0.473$, the structure coefficients are systematically larger than the pattern coefficients, particularly for items with substantial loadings on both factors. The further a point lies above the identity line, the more the factor intercorrelation is inflating the apparent relationship between that item and the factor.

The Rotation Objective Function Landscape

For two-factor orthogonal rotation, every rotation matrix is parameterized by a single angle $\theta \in [0, 2\pi)$:

$$T(\theta) = \begin{pmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{pmatrix}$$

This means the objective function f can be plotted as a function of θ , giving a complete picture of the rotation landscape — all local and global minima, and how flat or sharp they are.

The following function computes and plots $f(\theta)$ for n evenly spaced angles:

```
> plotRotationLandscape <- function(A, method = "quartimax", n = 20000,
                                   main = NULL, ...) {
  if (ncol(A) != 2)
    stop("plotRotationLandscape only works for 2-factor solutions.")

  vgQfun_fn <- get(paste("vgQ", method, sep = "."),
                   envir = asNamespace("GPArotation"))

  if (is.null(main))
    main <- paste("Rotation landscape:", method)

  theta <- seq(0, 2 * pi, length.out = n)
  f_vals <- numeric(n)

  for (i in seq_along(theta)) {
    Tmat <- matrix(c( cos(theta[i]), sin(theta[i]),
                     -sin(theta[i]), cos(theta[i])), 2, 2)
    L <- A %*% Tmat
    VgQ <- do.call(vgQfun_fn, append(list(L), list(...)))
    f_vals[i] <- VgQ$f
  }

  min_idx <- which.min(f_vals)

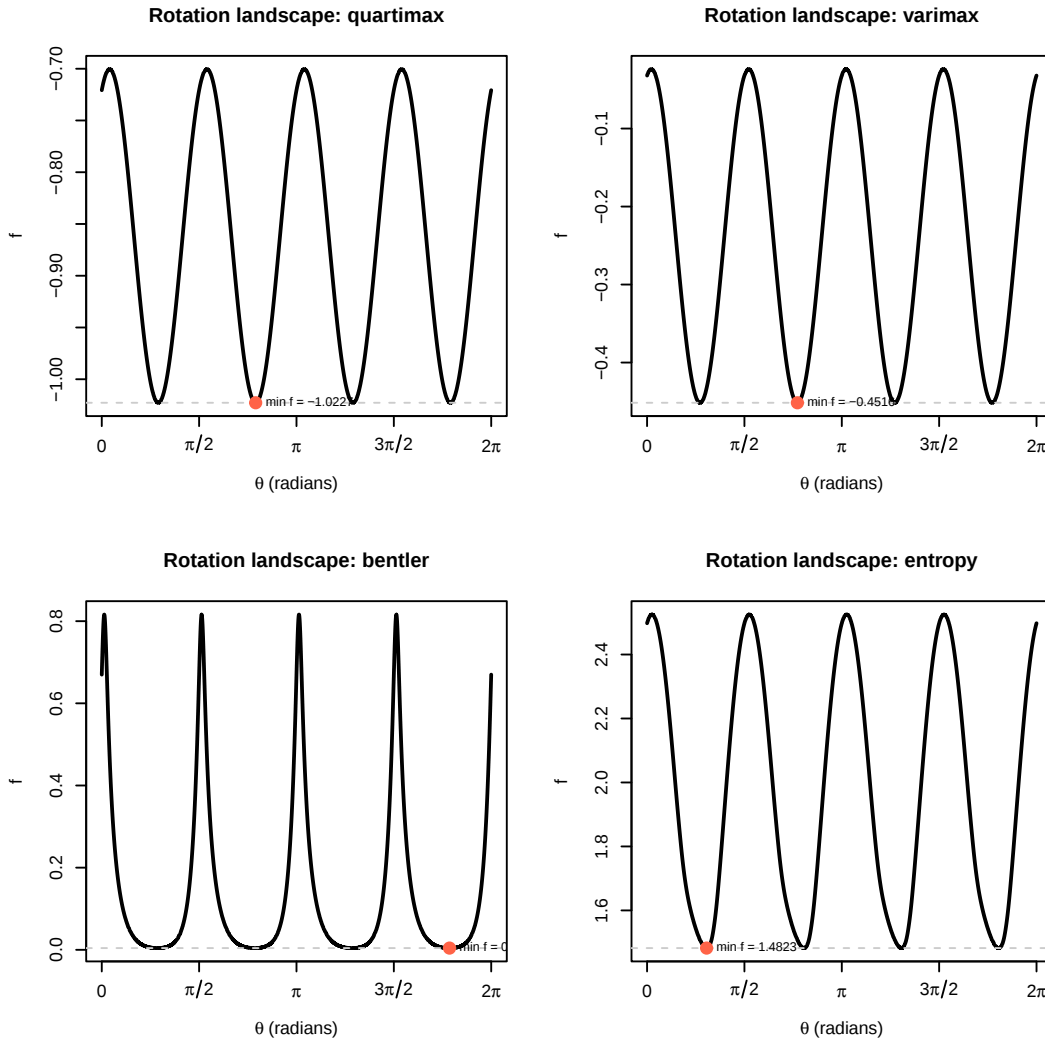
  plot(theta, f_vals,
       type = "l",
       lwd = 2,
       main = main,
       xlab = expression(theta ~ "(radians)"),
       ylab = "f",
       xaxt = "n")
  axis(1, at = c(0, pi/2, pi, 3*pi/2, 2*pi),
       labels = c("0", expression(pi/2), expression(pi),
                  expression(3*pi/2), expression(2*pi)))
  abline(h = f_vals[min_idx], col = "grey80", lty = 2)
  points(theta[min_idx], f_vals[min_idx],
        col = "tomato", pch = 19, cex = 1.5)
  text(theta[min_idx], f_vals[min_idx],
       labels = paste0("min f = ", round(f_vals[min_idx], 4)),
       pos = 4,
       cex = 0.8)

  invisible(data.frame(theta = theta, f = f_vals))
}
```

The following example compares four rotation criteria on the Harman 8-variable physical measurements dataset:

```
> data(Harman, package = "GPArotation")

> par(mfrow = c(2, 2))
> plotRotationLandscape(Harman8, method = "quartimax")
> plotRotationLandscape(Harman8, method = "varimax")
> plotRotationLandscape(Harman8, method = "bentler")
> plotRotationLandscape(Harman8, method = "entropy")
```



Interpreting the Landscape: Symmetry vs Genuine Local Minima

An important subtlety when interpreting rotation landscapes is that not all minima represent genuinely different factor structures. For a two-factor solution, the rotation landscape always contains at least 4 equivalent minima due to the inherent symmetry of factor analysis:

- **Sign flips:** flipping the sign of a factor column gives an equivalent solution. With two factors there are $2^2 = 4$ sign-flip combinations, each producing an equivalent minimum.
- **Column permutations:** swapping the two factors gives an equivalent solution, adding further symmetry.

Combined, a two-factor solution has at least 4 symmetry-equivalent minima in $[0, 2\pi)$. Any minima beyond these 4 represent genuinely different factor structures — true local minima in the optimization sense.

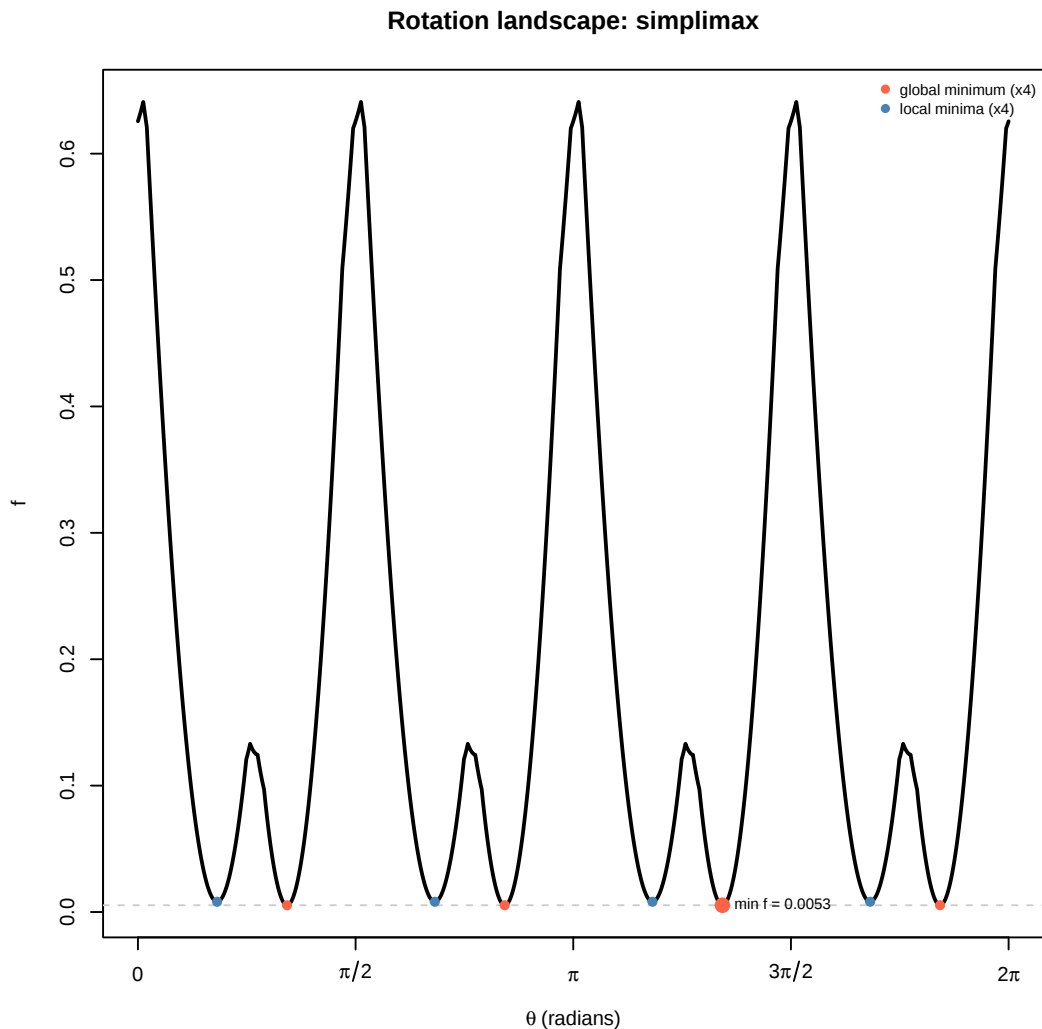
The following example illustrates this for the simplimax criterion with $k = 4$, which produces multiple local minima on the Harman 8-variable dataset:

```
> res <- plotRotationLandscape(Harman8, method = "simplimax", k = 4)
> # Find all local minima by sign changes in the discrete derivative
> df <- diff(res$f)
> local_mins <- which(df[-length(df)] < 0 & df[-1] > 0)
> # Classify minima: red = symmetry-equivalent global, blue = genuine local
> f_rounded <- round(res$f[local_mins], 4)
> f_min <- min(f_rounded)
> is_global <- f_rounded == f_min
```

```

> dot_cols <- ifelse(is_global, "tomato", "steelblue")
> n_global <- sum(is_global)
> n_local <- sum(!is_global)
> points(res$theta[local_mins], res$f[local_mins],
        col = dot_cols, pch = 19, cex = 0.9)
> legend("topright",
        legend = c(paste0("global minimum (x", n_global, ")"),
                    paste0("local minima (x", n_local, ")")),
        col = c("tomato", "steelblue"),
        pch = 19, bty = "n", cex = 0.8)

```



```

> cat("Total minima found:           ", length(local_mins), "\n")
Total minima found:                  8
> cat("Symmetry-equivalent global min: ", n_global, "\n")
Symmetry-equivalent global min:     4
> cat("Genuine local minima:          ", n_local, "\n")
Genuine local minima:                4

```

The `k` argument controls how many near-zero loadings simplimax targets. Smaller values of `k` create a rougher objective function landscape with more local minima, while larger values (up to `nrow(A)`) produce smoother landscapes.

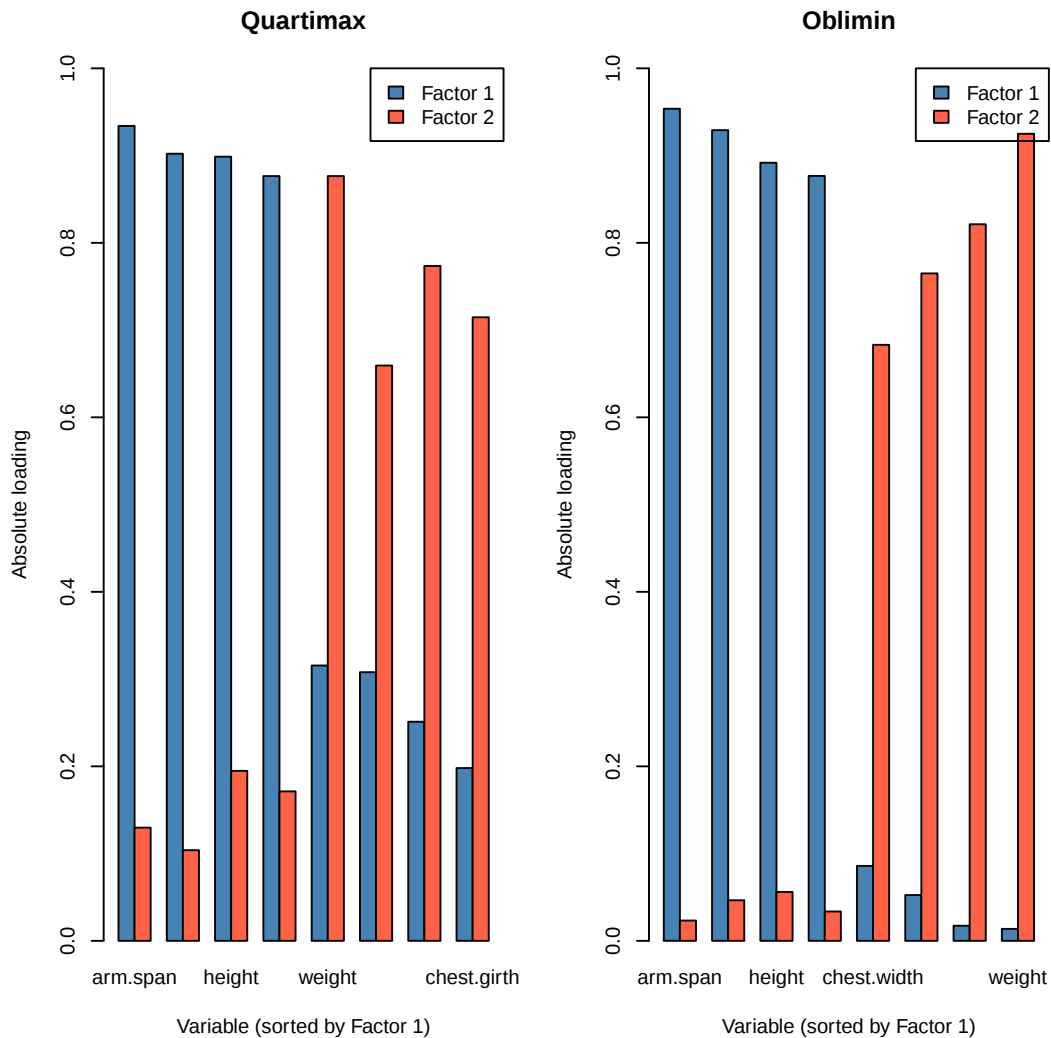
Criteria such as quartimax and varimax on well-structured data typically show exactly 4 minima — all symmetry-equivalent. Criteria prone to local minima, such as simplimax and geomin, show additional minima beyond the symmetry baseline. This is precisely why random starts are important: without them, the algorithm may converge to a symmetry-equivalent solution or a genuine local minimum rather than the global minimum.

The rotation landscape visualization is only available for two-factor solutions. For $k > 2$ factors the rotation space is $(k^2 - k)/2$ -dimensional and cannot be visualized as a simple curve. For higher-dimensional problems, the `GPFallMinima` function described in the `GPA2local` vignette provides an alternative approach to exploring the solution space.

Comparing Rotation Criteria Visually

Different rotation criteria can produce noticeably different loading patterns. A useful way to compare solutions is to make a bar graph of the absolute loadings sorted by magnitude for each factor. The following example compares quartimax and oblimin rotation of the Harman 8-variable physical measurements dataset.

```
> data(Harman, package = "GPArotation")
> res.quart <- quartimax(Harman8)
> res.oblimin <- oblimin(Harman8)
> L.quart <- abs(loadings(res.quart))
> L.oblimin <- abs(loadings(res.oblimin))
> ord.quart <- order(L.quart[, 1], decreasing = TRUE)
> ord.oblimin <- order(L.oblimin[, 1], decreasing = TRUE)
> par(mfrow = c(1, 2), mar = c(5, 4, 4, 2))
> barplot(t(L.quart[ord.quart, ]),
  beside = TRUE,
  ylim = c(0, 1),
  main = "Quartimax",
  ylab = "Absolute loading",
  xlab = "Variable (sorted by Factor 1)",
  legend.text = c("Factor 1", "Factor 2"),
  args.legend = list(x = "topright"),
  col = c("steelblue", "tomato"))
> barplot(t(L.oblimin[ord.oblimin, ]),
  beside = TRUE,
  ylim = c(0, 1),
  main = "Oblimin",
  ylab = "Absolute loading",
  xlab = "Variable (sorted by Factor 1)",
  legend.text = c("Factor 1", "Factor 2"),
  args.legend = list(x = "topright"),
  col = c("steelblue", "tomato"))
```



Quartimax tends to produce a general factor with high loadings on all variables, while oblimin allows factors to be correlated and typically produces a cleaner simple structure. The sorted absolute loading plot makes these differences immediately visible.

Sorted Absolute Loadings Plot

A useful way to compare the sparsity profile of different rotation solutions is to plot all absolute loadings sorted from smallest to largest. In a rotation with good simple structure, most loadings are near zero with a sharp upturn at the right — indicating that a few large loadings dominate. Comparing this profile across rotation criteria makes differences in simplicity and sparsity immediately visible.

The following function plots sorted absolute loadings for one or more `GPArotation` objects on a single figure.

```
> plotSortedLoadings <- function(..., labels = NULL, col = NULL,
  main = "Sorted Absolute Loadings",
  ylab = "Absolute loading",
  xlab = "Rank") {

  solutions <- list(...)

  for (i in seq_along(solutions)) {
    if (!inherits(solutions[[i]], "GPArotation"))
      stop("Argument ", i, " is not a GPArotation object.")
  }
}
```

```

n <- length(solutions)

if (is.null(labels))
  labels <- paste("Solution", seq_len(n))
if (is.null(col))
  col <- palette.colors(n, palette = "Okabe-Ito")

sorted_loadings <- lapply(solutions, function(x)
  sort(abs(as.vector(x$loadings)), decreasing = FALSE))

all_values <- unlist(sorted_loadings)
max_len <- max(sapply(sorted_loadings, length))

plot(NULL,
  xlim = c(1, max_len),
  ylim = c(0, max(all_values)),
  main = main,
  xlab = xlab,
  ylab = ylab,
  las = 1)

abline(h = seq(0, 1, by = 0.1), col = "grey90", lty = 1)

for (i in seq_len(n)) {
  lines(seq_along(sorted_loadings[[i]]), sorted_loadings[[i]],
    col = col[i], lwd = 2)
  points(seq_along(sorted_loadings[[i]]), sorted_loadings[[i]],
    col = col[i], pch = 19, cex = 0.6)
}

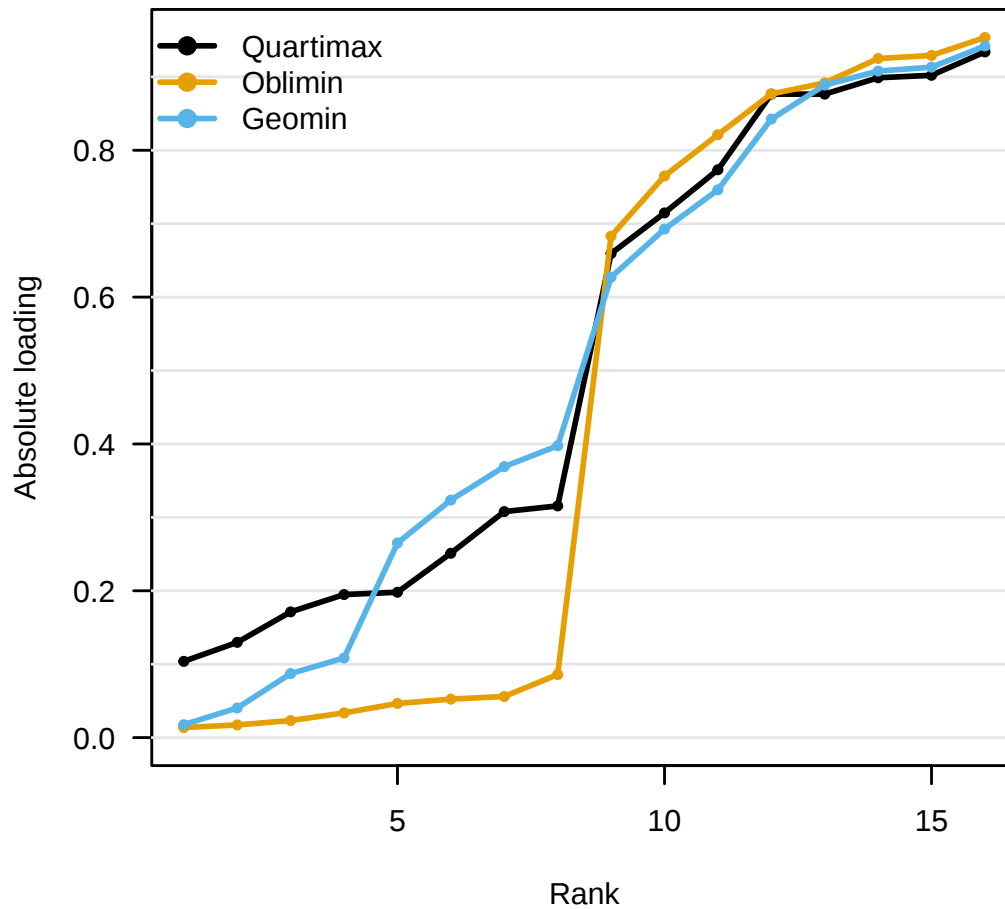
legend("topleft", legend = labels, col = col, lwd = 2, pch = 19,
  bty = "n")

invisible(sorted_loadings)
}
> # Example
> data(Harman, package = "GPArotation")
> res.quart <- quartimax(Harman8)
> res.oblimin <- oblimin(Harman8)
> res.geomin <- geominT(Harman8)

```

The following example compares quartimax, oblimin, and geomin rotation on the Harman 8-variable physical measurements dataset.

Sorted Absolute Loadings



A flat profile at the left indicates many near-zero loadings — a hallmark of simple structure. A gradual curve with no clear elbow suggests the criterion is not achieving strong separation between large and small loadings. Criteria that differ substantially in their sparsity assumptions, such as quartimax (which tends toward a general factor) versus oblimin (which allows correlated factors with cleaner simple structure), will produce visibly different profiles.

The function accepts any number of `GPArotation` objects and is useful for assessing the effect of different values of a criterion parameter, for example comparing rotation with different values of a parameter.

Special Rotation Types

An Example of Target Rotation

Fischer & Fontaine (2010) describe measuring self-reported extra-role behavior in samples of British and East German employees. They publish rotation matrices for two samples and investigate structural equivalence of the loadings matrices. The table lists the varimax rotated loadings matrices.

	Britain		East Germany	
	Factor 1	Factor 2	Factor 1	Factor 2
I am always punctual.	.783	-.163	.778	-.066
I do not take extra breaks.	.811	.202	.875	.081
I follow work rules and instructions with extreme care.	.724	.209	.751	.079
I never take long lunches or breaks.	.850	.064	.739	.092
I search for causes for something that did not function properly.	-.031	.592	.195	.574
I often motivate others to express their ideas and opinions.	-.028	.723	-.030	.807
During the last year I changed something in my work.	.388	.434	-.135	.717
I encourage others to speak up at meetings.	.141	.808	.125	.738
I continuously try to submit suggestions to improve my work.	.215	.709	.060	.691

The varimax rotations for each sample may be expected to be similar because the two loadings matrices are from different samples measuring the same constructs. Below are target rotation of the East German loadings matrix towards the British one, followed by calculation of agreement coefficients. Fischer & Fontaine (2010) note that coefficients generally should be “beyond the commonly accepted value of 0.90.”

```
> origdigits <- options("digits")
> options(digits = 2)
> trBritain <- matrix(c(.783,-.163,.811,.202,.724,.209,.850,.064,
  -.031,.592,-.028,.723,.388,.434,.141,.808,.215,.709),
  byrow = TRUE, ncol = 2)
> trGermany <- matrix(c(.778,-.066,.875,.081,.751,.079,.739,.092,
  .195,.574,-.030,.807,-.135,.717,.125,.738,.060,.691),
  byrow = TRUE, ncol = 2)
> # orthogonal rotation of trGermany towards trBritain
> trx <- targetT(trGermany, Target = trBritain)
> # Factor loadings after target rotation
> trx
```

Orthogonal rotation method Target rotation converged.

Loadings:

```
      [,1] [,2]
[1,] 0.774
[2,] 0.878
[3,] 0.754
[4,] 0.742
[5,] 0.219 0.565
[6,]      0.808
[7,] -0.104 0.722
[8,] 0.156 0.732
[9,]      0.688
```

```
      [,1] [,2]
SS loadings 2.58 2.52
Proportion Var 0.29 0.28
Cumulative Var 0.29 0.57
AUC          0.83 0.81
FSI          0.14 0.11
```

```
> # Differences between loadings matrices after rotation
> y <- trx$loadings - trBritain
> print(y, digits = 1)
```

Loadings:

```
      [,1] [,2]
```

```

[1,]
[2,]      -0.2
[3,]      -0.2
[4,]    -0.1
[5,]     0.3
[6,]
[7,]   -0.5  0.3
[8,]
[9,]   -0.1

      [,1] [,2]
SS loadings  0.3  0.2

> # Square root of the mean squared difference per item
> sqrt(apply((y^2), 1, mean))

[1] 0.046 0.122 0.117 0.076 0.178 0.064 0.403 0.055 0.090

> # Square root of the mean squared difference per factor
> sqrt(apply((y^2), 2, mean))

[1] 0.19 0.13

> # Identity coefficient per factor after rotation
> 2 * colSums(trx$loadings * trBritain) /
  (colSums(trx$loadings^2) + colSums(trBritain^2))

[1] 0.94 0.97

> # Additivity coefficient per factor after rotation
> diag(2 * cov(trx$loadings, trBritain)) /
  diag(var(trx$loadings) + var(trBritain))

[1] 0.86 0.92

> # Proportionality coefficient per factor after rotation
> colSums(trBritain * trx$loadings) /
  sqrt(colSums(trBritain^2) * colSums(trx$loadings^2))

[1] 0.94 0.97

> # Correlation for each factor after rotation
> diag(cor(trBritain, trx$loadings))

[1] 0.86 0.93

> options(digits = origdigits$digits)

```

The effect of target rotation can be visualized by plotting the factor loadings for both samples in the two-dimensional factor space. The following plot shows the British loadings (target), the German loadings before rotation, and the German loadings after rotation towards the British solution. Arrows connect each item's pre- and post-rotation position, making the movement induced by the rotation immediately visible.

```

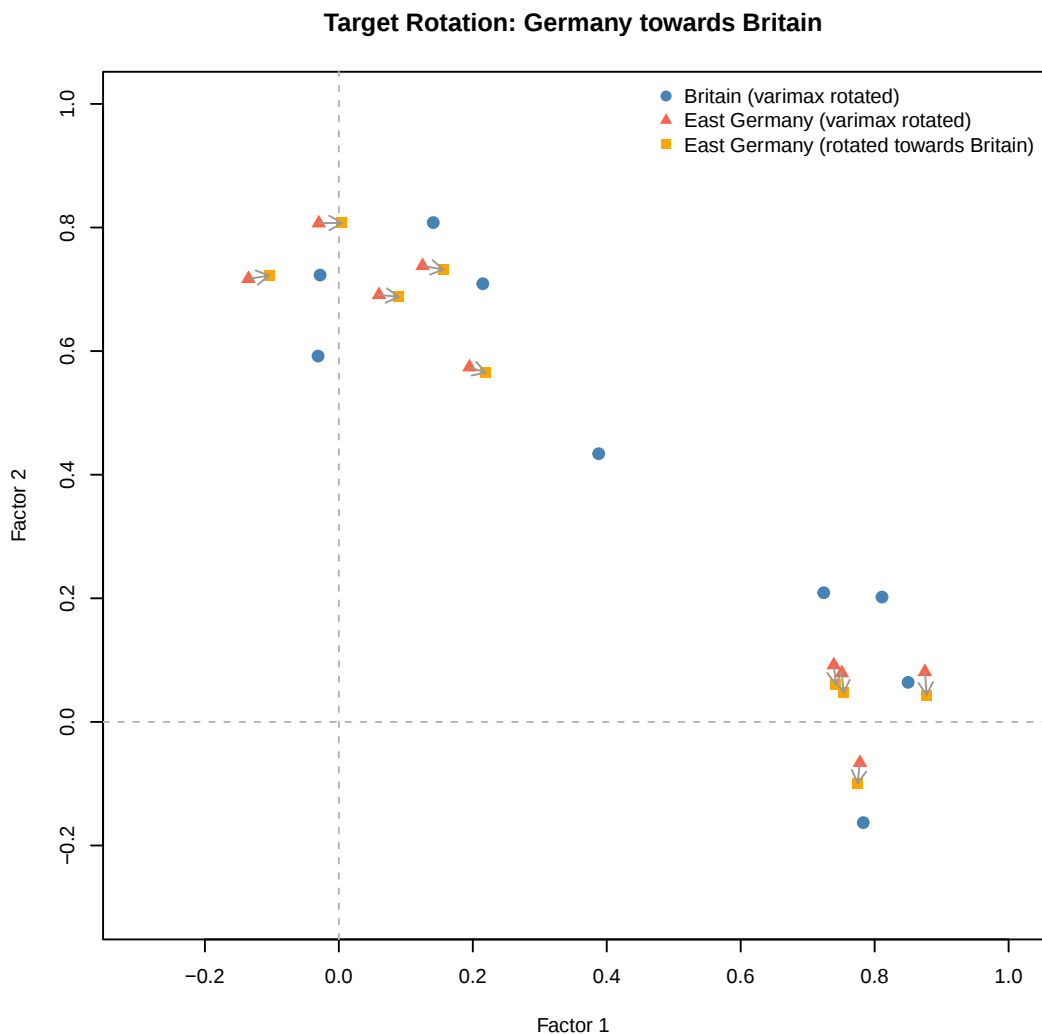
> plot(trBritain[, 1], trBritain[, 2],
      xlim = c(-0.3, 1.0), ylim = c(-0.3, 1.0),
      xlab = "Factor 1", ylab = "Factor 2",
      main = "Target Rotation: Germany towards Britain",
      pch = 19, col = "steelblue", cex = 1.2)
> abline(h = 0, lty = 2, col = "grey70")
> abline(v = 0, lty = 2, col = "grey70")
> points(trGermany[, 1], trGermany[, 2],

```

```

    pch = 17, col = "tomato", cex = 1.2)
> points(loadings(trx)[, 1], loadings(trx)[, 2],
    pch = 15, col = "orange", cex = 1.2)
> for (i in 1:nrow(trGermany)) {
    arrows(trGermany[i, 1], trGermany[i, 2],
    loadings(trx)[i, 1], loadings(trx)[i, 2],
    length = 0.08, col = "grey60")
}
> legend("topright",
    legend = c("Britain (varimax rotated)",
    "East Germany (varimax rotated)",
    "East Germany (rotated towards Britain)"),
    col = c("steelblue", "tomato", "orange"),
    pch = c(19, 17, 15),
    bty = "n")

```



Items whose arrows are short moved little during rotation, indicating that the German and British loadings were already similar for those items. Items with longer arrows required more adjustment, suggesting greater cultural differences in how those behaviors are structured. After rotation, the German loadings may be compared directly to the British target using the agreement coefficients computed above.

An Example of Partially Specified Target Rotation

Browne (1972) reported an initial loadings matrix and a partially specified target to rotate towards. In `GPArotation` the partially specified target matrix is of the same dimension as the initial matrix `A`, with `NA` in entries that are not pre-specified. Both target rotation and partially specified target rotation can be used to reproduce Browne (1972) results.

In this orthogonal rotation example, `targetT` includes a `Target` matrix with `NA` in entries not used in target rotation. With `pstT` no missing values are present in the `Target` matrix, and the weight matrix `W` includes weight 0 for entries not used, and 1 for entries included in the rotation.

```
> A <- matrix(c(.664, .688, .492, .837, .705, .82, .661, .457, .765, .322,
               .248, .304, -0.291, -0.314, -0.377, .397, .294, .428, -0.075, .192, .224,
               .037, .155, -.104, .077, -.488, .009), ncol = 3)
> # using targetT
> SPA <- matrix(c(rep(NA, 6), .7, .0, .7, rep(0, 3), rep(NA, 7),
                 0, 0, NA, 0, rep(NA, 4)), ncol = 3)
> xt <- targetT(A, Target = SPA)
> # using pstT
> SPApst <- matrix(c(rep(0, 6), .7, .0, .7, rep(0, 3), rep(0, 7),
                   0, 0, 0, 0, rep(0, 4)), ncol = 3)
> SPAW <- matrix(c(rep(0, 6), rep(1, 6), rep(0, 7), 1, 1, 0, 1,
                   rep(0, 4)), ncol = 3)
> xpst <- pstT(A, Target = SPApst, W = SPAW)
> max(abs(loadings(xt) - loadings(xpst)))

[1] 0
```

Note that convergence tables are identical for both methods. Additional examples are available in the help pages of `GPfoblq` and `rotations`.

Using GPArotation with factanal

Passing Rotation to factanal

Rotation functions can be passed directly to `factanal` via the `rotation` argument. Additional arguments are passed through the control list.

```
> data("CCAI", package = "GPArotation")
> factanal(factors = 3, covmat = CCAI_R, n.obs = 461, rotation = "infomaxT")
```

Call:

```
factanal(factors = 3, covmat = CCAI_R, n.obs = 461, rotation = "infomaxT")
```

Uniquenesses:

CCAI8	CCAI6	CCAI7	CCAI11	CCAI12	CCAI10	CCAI14	CCAI13	CCAI5	CCAI2	CCAI4
0.128	0.272	0.299	0.247	0.266	0.347	0.055	0.077	0.286	0.612	0.321
CCAI1	CCAI3	CCAI9								
0.455	0.364	0.372								

Loadings:

	Factor1	Factor2	Factor3
CCAI8	0.864	0.277	0.222
CCAI6	0.774	0.289	0.214
CCAI7	0.744	0.343	0.172
CCAI11	0.634	0.434	0.404
CCAI12	0.600	0.397	0.465
CCAI10	0.520	0.463	0.410
CCAI14	0.302	0.277	0.882
CCAI13	0.318	0.313	0.851

CCAI5	0.308	0.460	0.638
CCAI2	0.237	0.567	0.102
CCAI4	0.273	0.739	0.241
CCAI1	0.278	0.656	0.193
CCAI3	0.355	0.653	0.290
CCAI9	0.409	0.589	0.337

	Factor1	Factor2	Factor3
SS loadings	3.72	3.296	2.884

Test of the hypothesis that 3 factors are sufficient.
The chi square statistic is 387.64 on 52 degrees of freedom.
The p-value is 7.58e-53

```
> factanal(factors = 3, covmat = CCAI_R, n.obs = 461, rotation = "infomaxT",
  control = list(rotate = list(normalize = TRUE, eps = 1e-6)))
```

Call:

```
factanal(factors = 3, covmat = CCAI_R, n.obs = 461, rotation = "infomaxT", control = list(rotate = list
```

Uniquenesses:

CCAI8	CCAI6	CCAI7	CCAI11	CCAI12	CCAI10	CCAI14	CCAI13	CCAI5	CCAI2	CCAI4
0.128	0.272	0.299	0.247	0.266	0.347	0.055	0.077	0.286	0.612	0.321
CCAI1	CCAI3	CCAI9								
0.455	0.364	0.372								

Loadings:

	Factor1	Factor2	Factor3
CCAI8	0.322	0.852	0.205
CCAI6	0.329	0.762	0.198
CCAI7	0.381	0.729	0.155
CCAI11	0.473	0.617	0.386
CCAI12	0.436	0.586	0.448
CCAI10	0.496	0.502	0.392
CCAI14	0.312	0.298	0.871
CCAI13	0.348	0.312	0.840
CCAI5	0.489	0.293	0.623
CCAI2	0.580	0.211	
CCAI4	0.756	0.241	0.220
CCAI1	0.673	0.249	0.174
CCAI3	0.676	0.327	0.270
CCAI9	0.615	0.385	0.318

	Factor1	Factor2	Factor3
SS loadings	3.671	3.507	2.721

Test of the hypothesis that 3 factors are sufficient.
The chi square statistic is 387.64 on 52 degrees of freedom.
The p-value is 7.58e-53

Two-Step Procedure (Generally Recommended)

For oblique rotation, the recommended approach is to obtain unrotated loadings from `factanal` and rotate separately using `GPARotation`. This gives full control over the rotation, including random starts, algorithm selection, and convergence diagnostics. `GPARotation` accepts `factanal` objects directly — no need to extract loadings manually.

```
> data("CCAI", package = "GPARotation")
> fa.un <- factanal(factors = 3, covmat = CCAI_R, n.obs = 461,
```

```

      rotation = "none")
> oblimin(fa.un, randomStarts = 100)

Oblique rotation method Oblimin Quartimin converged at lowest minimum.
Of 100 random starts 100% converged, 100% at the same lowest minimum.
Loadings at lowest minimum:

```

	Factor1	Factor2	Factor3
CCAI8	0.984		
CCAI6	0.858		
CCAI7	0.806	0.100	
CCAI11	0.537	0.198	0.237
CCAI12	0.489	0.138	0.338
CCAI10	0.366	0.287	0.263
CCAI14			1.001
CCAI13			0.946
CCAI5		0.296	0.619
CCAI2		0.668	-0.107
CCAI4		0.848	
CCAI1		0.745	
CCAI3		0.672	
CCAI9	0.179	0.541	0.154

	Factor1	Factor2	Factor3
SS loadings	3.601	3.260	3.037
AUC	0.888	0.855	0.920
FSI	0.160	0.118	0.245

Phi:

	Factor1	Factor2	Factor3
Factor1	1.000	0.707	0.622
Factor2	0.707	1.000	0.652
Factor3	0.622	0.652	1.000

Prior to R 4.5.1, the single-step approach (rotation inside `factanal`) had a bug in factor reordering after oblique rotation, reported by Bernaards and others and fixed by the R core team. The two-step procedure has always been correct regardless of R version. Users on $R \geq 4.5.1$ may use either approach; the two-step procedure is still preferred for access to random starts and convergence diagnostics.

The following example verifies agreement between the two approaches using a non-standard Crawford-Ferguson kappa value:

```

> data("WansbeekMeijer", package = "GPARotation")
> fa.un <- factanal(factors = 3, covmat = NetherlandsTV,
+                  normalize = TRUE, rotation = "none")
> set.seed(42)
> res.cf <- cfQ(fa.un, kappa = 0.3, normalize = TRUE, randomStarts = 100)
> res.cf

```

Oblique rotation method Crawford-Ferguson: kappa=0.3 converged at lowest minimum.
Of 100 random starts 100% converged, 100% at the same lowest minimum.
Loadings at lowest minimum:

	Factor1	Factor2	Factor3
NL1	0.713	0.111	
TV2	0.700	0.131	0.162
NL3	0.692		
RTL4		0.107	0.618
RTL5		0.265	0.525
Veronica		0.251	0.642
SBS6		1.066	

	Factor1	Factor2	Factor3
SS loadings	1.672	1.569	1.435
AUC	0.855	0.957	0.857
FSI	0.213	0.707	0.200

Phi:

	Factor1	Factor2	Factor3
Factor1	1.000	0.313	0.392
Factor2	0.313	1.000	0.623
Factor3	0.392	0.623	1.000

```
> if (getRversion() >= "4.5.1") {
  set.seed(42)
  fa.cf <- factanal(factors = 3, covmat = NetherlandsTV,
                    normalize = TRUE, rotation = "cfQ",
                    control = list(rotate = list(kappa = 0.3,
                                                  randomStarts = 100)))

  cat("Maximum difference in loadings:\n")
  print(max(abs(abs(res.cf$loadings) - abs(fa.cf$loadings))))
} else {
  cat("Use the two-step procedure above for correct results.\n")
}
```

Maximum difference in loadings:
[1] 1.033882

Rotation Criteria Reference

The following table lists all rotation criteria available in **GPArotation**, with their type and key reference. Criteria ending in T are orthogonal; those ending in Q are oblique. Criteria without a T/Q suffix may be used for both.

Function	Type	Criterion
oblimin	oblique	Oblimin family; gam controls obliqueness
quartimin	oblique	Oblimin with gam = 0
targetT	orthogonal	Rotation towards a target matrix
targetQ	oblique	Rotation towards a target matrix
pstT	orthogonal	Partially specified target rotation
pstQ	oblique	Partially specified target rotation
oblimax	oblique	Maximizes overall kurtosis of loadings
entropy	orthogonal	Minimizes entropy of squared loadings
quartimax	orthogonal	Maximizes variance of squared loadings within variables
Varimax	orthogonal	Maximizes variance of squared loadings within factors
simplimax	oblique	Minimizes the k smallest squared loadings
bentlerT	orthogonal	Invariant pattern simplicity
bentlerQ	oblique	Invariant pattern simplicity
tandemI	orthogonal	Factors share high loadings on same variables
tandemII	orthogonal	Factors do not share high loadings on same variables
geominT	orthogonal	Minimizes geometric mean of squared loadings
geominQ	oblique	Minimizes geometric mean of squared loadings
bigominT	orthogonal	Geomin with a general factor in column 1
bigominQ	oblique	Geomin with a general factor in column 1
cft	orthogonal	Crawford-Ferguson family; kappa controls complexity
cfQ	oblique	Crawford-Ferguson family; kappa controls complexity
equamax	orthogonal	Crawford-Ferguson with $\kappa = m/(2p)$
parsimax	orthogonal	Crawford-Ferguson with $\kappa = (m - 1)/(p + m - 2)$
infomaxT	orthogonal	Infomax information criterion
infomaxQ	oblique	Infomax information criterion
mccammon	orthogonal	Minimizes entropy ratio across factors
varimin	orthogonal	Minimizes variance of squared loadings within factors
bifactorT	orthogonal	Bifactor; general factor in column 1
bifactorQ	oblique	Biquartimin; general factor in column 1
lpT	orthogonal	L^p sparsity rotation
lpQ	oblique	L^p sparsity rotation

Cureton-Mulaik Normalization

Row normalization rescales the rows of the loading matrix before rotation and back-transforms after. The most common normalization is Kaiser (Horst) normalization (**normalize** = **TRUE**), which scales each row to unit norm, giving equal weight to all items regardless of communality. Cureton & Mulaik (1975) proposed an alternative weighting scheme that assigns weights based on the angular distance of each item from the equal-loading point on the unit hypersphere. Items with clear simple structure receive low weight; items with ambiguous cross-loadings receive high weight.

This sounds appealing but Browne (2001) showed it can fail badly. Consider the constructed matrix from his Table 5, where items 1–10 have clean simple structure (loading on exactly one factor) and items 11–12 are cross-loaders:

```
> Lambda <- matrix(c(
  0.90, 0.00, 0.00,
  0.90, 0.00, 0.00,
  0.90, 0.00, 0.00,
  0.90, 0.00, 0.00,
  0.00, 0.80, 0.00,
  0.00, 0.80, 0.00,
  0.00, 0.80, 0.00,
  0.00, 0.00, 0.70,
  0.00, 0.00, 0.70,
  0.00, 0.00, 0.70,
  0.10, 0.10, 0.00,
  0.10, 0.00, 0.10
), nrow = 12, ncol = 3, byrow = TRUE,
```

```

dimnames = list(paste0("Var", 1:12), paste0("Fac", 1:3))
> h      <- sqrt(rowSums(Lambda^2))
> kaiser_wts <- round(1 / h, 2)
> al_cm    <- GPArotation::NormalizingWeight(Lambda,
                                              normalize = "CM")[, 1]
> cm_wts   <- round(pmin(pmax(h / al_cm, 0), 1), 2)
> print(cbind(round(Lambda, 2),
              "K Wts." = kaiser_wts,
              "CM Wts." = cm_wts))

```

	Fac1	Fac2	Fac3	K Wts.	CM Wts.
Var1	0.9	0.0	0.0	1.11	0.00
Var2	0.9	0.0	0.0	1.11	0.00
Var3	0.9	0.0	0.0	1.11	0.00
Var4	0.9	0.0	0.0	1.11	0.00
Var5	0.0	0.8	0.0	1.25	0.00
Var6	0.0	0.8	0.0	1.25	0.00
Var7	0.0	0.8	0.0	1.25	0.00
Var8	0.0	0.0	0.7	1.43	0.00
Var9	0.0	0.0	0.7	1.43	0.00
Var10	0.0	0.0	0.7	1.43	0.00
Var11	0.1	0.1	0.0	7.07	0.92
Var12	0.1	0.0	0.1	7.07	0.92

Items 1–10 — the items that define simple structure — receive a CM weight of exactly zero. Items 11–12 — the ambiguous cross-loaders — receive the only non-zero weights. The Cureton-Mulaik rotation is therefore driven entirely by the two least informative items, producing a meaningless result:

```

> rot_cm <- GPForth(Lambda, method = "varimax", normalize = "CM")
> print(round(unclass(rot_cm$loadings), 2))

```

	Fac1	Fac2	Fac3
Var1	0.52	0.52	0.52
Var2	0.52	0.52	0.52
Var3	0.52	0.52	0.52
Var4	0.52	0.52	0.52
Var5	-0.46	0.63	-0.17
Var6	-0.46	0.63	-0.17
Var7	-0.46	0.63	-0.17
Var8	-0.40	-0.15	0.55
Var9	-0.40	-0.15	0.55
Var10	-0.40	-0.15	0.55
Var11	0.00	0.14	0.04
Var12	0.00	0.04	0.14

```

attr("covariance")
      [,1] [,2] [,3]
[1,]    1    0    0
[2,]    0    1    0
[3,]    0    0    1

```

Vars 1–4 all receive identical loadings of 0.52 on all three factors — the rotation has completely failed to recover the simple structure. This is the failure mode documented by Browne (2001): when items with near-perfect simple structure have very low communalities relative to cross-loaders, Cureton-Mulaik normalization suppresses exactly the items that should drive the rotation.

0.1 Effect of Interfactor Correlation

Browne (2001) also showed that CM normalization performs better when factors are oblique. Table 6 in his paper examines oblique CF-Varimax with CM normalization across three levels of interfactor correlation:

```

> rhos      <- c(0.0, 0.3, 0.5)
> patterns <- list()
> for (rho in rhos) {
  Phi_true   <- matrix(rho, 3, 3); diag(Phi_true) <- 1
  R_common   <- Lambda %%% Phi_true %%% t(Lambda)
  ev          <- eigen(R_common)
  A           <- ev$vectors[, 1:3] %%% diag(sqrt(ev$values[1:3]))
  rownames(A) <- paste0("Var", 1:12)
  rot         <- GPFoblq(A, method = "cf",
                        methodArgs = list(kappa = 1/12),
                        normalize   = "CM")

  P           <- unclass(rot$loadings)
  col_order   <- order(colSums(P^2), decreasing = TRUE)
  P           <- P[, col_order]
  for (j in 1:3) if (sum(P[, j]) < 0) P[, j] <- -P[, j]
  colnames(P) <- paste0("Fac", 1:3)
  patterns[[as.character(rho)]] <- round(P, 2)
}
> t6 <- cbind(patterns[["0.0"]], patterns[["0.3"]], patterns[["0.5"]])
> colnames(t6) <- c("rho=0: F1", "F2", "F3",
                   "rho=.3: F1", "F2", "F3",
                   "rho=.5: F1", "F2", "F3")
> print(t6)

```

	rho=0: F1	F2	F3	rho=.3: F1	F2	F3	rho=.5: F1	F2	F3
Var1	-1.12	-0.15	-0.15	0.97	-0.08	-0.15	0.94	-0.03	-0.06
Var2	-1.12	-0.15	-0.15	0.97	-0.08	-0.15	0.94	-0.03	-0.06
Var3	-1.12	-0.15	-0.15	0.97	-0.08	-0.15	0.94	-0.03	-0.06
Var4	-1.12	-0.15	-0.15	0.97	-0.08	-0.15	0.94	-0.03	-0.06
Var5	1.00	1.27	0.13	0.05	0.78	0.02	0.06	0.76	0.03
Var6	1.00	1.27	0.13	0.05	0.78	0.02	0.06	0.76	0.03
Var7	1.00	1.27	0.13	0.05	0.78	0.02	0.06	0.76	0.03
Var8	0.87	0.12	1.10	0.05	0.02	0.68	0.06	0.04	0.66
Var9	0.87	0.12	1.10	0.05	0.02	0.68	0.06	0.04	0.66
Var10	0.87	0.12	1.10	0.05	0.02	0.68	0.06	0.04	0.66
Var11	0.00	0.14	0.00	0.11	0.09	-0.01	0.11	0.09	0.00
Var12	0.00	0.00	0.14	0.11	-0.01	0.08	0.11	0.00	0.09

At $\rho = 0$ (orthogonal factors) the CM-weighted rotation produces a degenerate solution — the criterion surface is flat because items 1–10 receive zero weight (as shown above), leaving the rotation driven only by the two cross-loading items. The specific degenerate solution found may vary across implementations and platforms. At $\rho = 0.3$ and $\rho = 0.5$ the solution recovers cleanly: primary loadings are near their true values and cross-loadings are near zero, confirming that CM normalization can be appropriate for oblique rotation when factors are moderately to strongly correlated.

The practical conclusion: `normalize = "CM"` may be appropriate for oblique rotation of data with moderate to high factor intercorrelations, but should be used with caution for orthogonal rotation or data with extreme communality differences across items. The default `normalize = TRUE` (Kaiser) is more robust across conditions.

- `normalize = FALSE` — no normalization (default)
- `normalize = TRUE` or `normalize = "Kaiser"` — Kaiser (Horst) normalization; scales rows to unit norm
- `normalize = "CM"` — Cureton-Mulaik normalization; use with caution (see Browne 2001)
- `normalize = v` — custom weight vector of length p
- `normalize = f` — custom function taking A and returning a weight vector

Further Resources

Full documentation for all functions is available via the R help system, for example `?quartimax` or `?GPFORSorth`. The package index is accessible via `?GPARotation`.

The following vignettes are provided with **GPArotation**:

- `vignette("GPA2local", package = "GPArotation")` — assessing local minima in factor rotation, including the `GPFallMinima` function and sorted loadings plots
- `vignette("GPA3bifactor", package = "GPArotation")` — bifactor rotation and reliability coefficients including omega hierarchical

Gradient projection *without* derivatives can be performed using the **GPArotatedF** package, available separately on CRAN. A vignette is provided with that package; type `vignette("GPArotatedF", package = "GPArotatedF")` at the R prompt.

For detailed investigation of local minima in factor rotation, the following packages provide complementary functionality:

- *fungible*: `faMain` function with extensive random start diagnostics
- *psych*: `faRotations` function for rotation comparison

References for Rotation Criteria

The following references describe the theoretical basis for each rotation criterion implemented in **GPArotation**.

Descriptions of many rotation criteria

Browne, M.W. (2001). An overview of analytic rotation in exploratory factor analysis. *Multivariate Behavioral Research*, **36**, 111–150. doi: 10.1207/S15327906MBR3601_05

Harman, H.H. (1976). *Modern Factor Analysis* (3rd ed.). The University of Chicago Press.

Oblimin / Quartimin

Carroll, J.B. (1960). IBM 704 program for generalized analytic rotation solution in factor analysis. Harvard University, unpublished.

Jennrich, R.I. (1979). Admissible values of γ in direct oblimin rotation. *Psychometrika*, **44**, 173–177. doi: 10.1007/BF02293969

Target rotation

Harman, H.H. (1976). *Modern Factor Analysis* (3rd ed.). The University of Chicago Press.

Partially specified target rotation

Browne, M.W. (1972a). Orthogonal rotation to a partially specified target. *British Journal of Mathematical and Statistical Psychology*, **25**, 115–120. doi: 10.1111/j.2044-8317.1972.tb00482.x

Browne, M.W. (1972b). Oblique rotation to a partially specified target. *British Journal of Mathematical and Statistical Psychology*, **25**, 207–212. doi: 10.1111/j.2044-8317.1972.tb00492.x

Oblimax

Pinzka, C. and Saunders, D.R. (1954). Analytic rotation to simple structure: II. Extension to an oblique solution. Research Bulletin 54–31. Princeton, N.J.: Educational Testing Service.

Saunders, D.R. (1961). The rationale for an “oblimax” method of transformation in factor analysis. *Psychometrika*, **26**, 317–324. doi: 10.1007/BF02289800

Minimum entropy

Jennrich, R.I. (2004). Rotation to simple loadings using component loss functions: the orthogonal case. *Psychometrika*, **69**, 257–274. doi: 10.1007/BF02295943

Quartimax

- Carroll, J.B. (1953). An analytic solution for approximating simple structure in factor analysis. *Psychometrika*, **18**, 23–38. doi: 10.1007/BF02289025
- Ferguson, G.A. (1954). The concept of parsimony in factor analysis. *Psychometrika*, **19**, 281–290. doi: 10.1007/BF02289228
- Neuhaus, J.O. and Wrigley, C. (1954). The quartimax method: An analytical approach to orthogonal simple structure. *British Journal of Statistical Psychology*, **7**, 81–91. doi: 10.1111/j.2044-8317.1954.tb00147.x

Varimax

- Kaiser, H.F. (1958). The varimax criterion for analytic rotation in factor analysis. *Psychometrika*, **23**, 187–200. doi: 10.1007/BF02289233

Simplimax

- Kiers, H.A.L. (1994). SIMPLIMAX: Oblique rotation to an optimal target with simple structure. *Psychometrika*, **59**, 567–579. doi: 10.1007/BF02294392

Bentler invariant pattern simplicity

- Bentler, P.M. (1977). Factor simplicity index and transformations. *Psychometrika*, **42**, 277–295. doi: 10.1007/BF02294054

Tandem criteria

- Comrey, A.L. (1967). Tandem criteria for analytic rotation in factor analysis. *Psychometrika*, **32**, 277–295. doi: 10.1007/BF02289422

Geomin

- Yates, A. (1984). *Multivariate Exploratory Data Analysis: A Perspective on Exploratory Factor Analysis*. State University of New York Press.

Bi-Geomin

- Garcia-Garzon, E., Abad, F.J., and Garrido, L.E. (2021). On omega hierarchical estimation: A comparison of exploratory bi-factor analysis algorithms. *Multivariate Behavioral Research*, **56**(1), 101–119. doi: 10.1080/00273171.2020.1736977

Crawford-Ferguson family

- Crawford, C.B. and Ferguson, G.A. (1970). A general rotation criterion and its use in orthogonal rotation. *Psychometrika*, **35**, 321–332. doi: 10.1007/BF02310572

Infomax

- McKeon, J.J. (1968). Rotation for maximum association between factors and tests. Unpublished manuscript, Biometric Laboratory, George Washington University.

McCammon minimum entropy ratio

- McCammon, R.B. (1966). Principal components analysis and its application in large-scale correlation studies. *Journal of Geology*, **74**, 721–733. doi: 10.1086/627207

Varimin

- Ertel, S. (2011). Exploratory factor analysis revealing complex structure. *Personality and Individual Differences*, **50**(2), 196–200. doi: 10.1016/j.paid.2010.09.026

Bifactor

- Jennrich, R.I. and Bentler, P.M. (2011). Exploratory bi-factor analysis. *Psychometrika*, **76**(4), 537–549. doi: 10.1007/s11336-011-9218-4

Lp rotation

Liu, X., Wallin, G., Chen, Y., and Moustaki, I. (2023). Rotation to sparse loadings using L^p losses and related inference problems. *Psychometrika*, **88**(2), 527–553. doi: 10.1007/s11336-023-09911-y

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. <https://doi.org/10.1177/0013164404272507>
- Browne, M. W. (1972). Orthogonal rotation to a partially specified target. *British Journal of Mathematical and Statistical Psychology*, 25(1), 115–120. doi: 10.1111/j.2044-8317.1972.tb00482.x
- Browne, M.W. (2001). An overview of analytic rotation in exploratory factor analysis. *Multivariate Behavioral Research*, **36**(1), 111–150. doi: 10.1207/S15327906MBR3601_05
- Cureton, E.E. and Mulaik, S.A. (1975). The weighted varimax rotation and the promax rotation. *Psychometrika*, **40**(2), 183–195. doi: 10.1007/BF02291565
- Fischer, R., & Fontaine, J. (2010). Methods for investigating structural equivalence. In D. Matsumoto, & F. van de Vijver (Eds.), *Cross-Cultural Research Methods in Psychology* (pp. 179–215). Cambridge University Press. doi: 10.1017/CBO9780511779381.010
- Mansolf, M. and Reise, S.P. (2016). Exploratory bifactor analysis: The Schmid-Leiman orthogonalization and Jennrich-Bentler analytic rotations. *Multivariate Behavioral Research*, **51**(5), 698–717. doi: 10.1080/00273171.2016.1215898
- Nguyen, H. V., & Waller, N. G. (2022). Local minima and factor rotations in exploratory factor analysis. *Psychological Methods*. Advance online publication. doi: 10.1037/met0000467