# Assessing Local Minima in Factor Rotation

## The `GPFallMinima` Function

Author: Coen A. Bernaards

Many rotation criteria have multiple local minima. The standard `GPArotation` functions return only the best solution found among all random starts — the one with the lowest criterion value. While this is the right default behavior, it conceals potentially important information: how many distinct solutions exist, how often each is found, and how different they are from each other.

This vignette introduces `GPFallMinima`, a companion function that runs multiple random starts and retains *all* distinct local minima, not just the global minimum. This allows the analyst to assess the stability of the rotation solution and to examine alternative solutions that may be substantively meaningful.

For background on local minima in factor rotation, see Nguyen & Waller (2022) and Mansolf & Reise (2016).

## The GPFallMinima Function

`GPFallMinima` is not part of the standard `GPArotation` package. It is a companion utility intended for diagnostic use. Load it alongside `GPArotation`:

```
> library("GPArotation")
> GPFallMinima <- function(A, method = "quartimin", orthogonal = FALSE,
                           randomStarts = 100, eps = 1e-5, maxit = 1000,
                           normalize = FALSE, methodArgs = NULL,
                           minimumInclusion = 2) {
    # Runs multiple random starts and returns ALL distinct local minima,
    # not just the global minimum. Non-converged starts are discarded.
    # Minima found fewer than minimumInclusion times are excluded.
    # Minima are sorted by frequency (most common first).

    engine <- if (orthogonal) GPForth else GPFoblq

    Qvalues    <- numeric(randomStarts)
    Qconverged <- logical(randomStarts)
    all_res    <- vector("list", randomStarts)

    for (i in 1:randomStarts) {
      res <- engine(A, Tmat      = Random.Start(ncol(A)),
                    normalize  = normalize,
                    eps        = eps,
                    maxit      = maxit,
                    method     = method,
                    methodArgs = methodArgs)
```

```
  Qvalues[i]    <- res$Table[nrow(res$Table), 2]
  Qconverged[i] <- res$convergence
  all_res[[i]]  <- res
}

# Discard non-converged starts
converged_idx <- which(Qconverged)
nConverged    <- length(converged_idx)

if (nConverged == 0)
  stop("No starts converged. Consider increasing maxit or relaxing eps.")

Qvalues_conv <- Qvalues[converged_idx]
all_res_conv <- all_res[converged_idx]

# Bin converged criterion values into equivalence classes
Q_round  <- round(Qvalues_conv / eps) * eps
Q_unique <- unique(Q_round)

# Build one representative solution per unique minimum
minima <- vector("list", length(Q_unique))
for (j in seq_along(Q_unique)) {
  idx        <- which(Q_round == Q_unique[j])
  count      <- length(idx)
  proportion <- count / nConverged
  minima[[j]] <- list(
    result     = all_res_conv[[idx[1]]],
    f          = Q_unique[j],
    count      = count,
    proportion = proportion
  )
}

# Sort by count (most common first)
ord    <- order(sapply(minima, `[[`, "count"), decreasing = TRUE)
minima <- minima[ord]

# Filter out minima found fewer than minimumInclusion times
keep   <- sapply(minima, `[[`, "count") >= minimumInclusion
minima <- minima[keep]

if (length(minima) == 0)
  stop("No minima found with count >= minimumInclusion (", minimumInclusion,
       "). Consider reducing minimumInclusion or increasing randomStarts.")

# Summary data frame
```

```
    f_values <- sapply(minima, `[[`, "f")
    f_global <- min(f_values)

    summary_df <- data.frame(
      minimum    = seq_along(minima),
      f          = round(f_values, 6),
      deltaF     = round(f_values - f_global, 6),
      count      = sapply(minima, `[[`, "count"),
      proportion = round(sapply(minima, `[[`, "proportion"), 3),
      isGlobal   = f_values == f_global
    )

    result <- list(
      minima           = minima,
      summary          = summary_df,
      Qvalues          = Qvalues_conv,
      nConverged       = nConverged,
      nStarts          = randomStarts,
      method           = method,
      orthogonal       = orthogonal,
      minimumInclusion = minimumInclusion
    )
    class(result) <- "GPFallMinima"
    result
  }
> print.GPFallMinima <- function(x, ...) {
    cat("Random start analysis:", x$nConverged, "of", x$nStarts,
        "starts converged\n")
    cat("Distinct minima found:", nrow(x$summary),
        "(minimumInclusion =", x$minimumInclusion, ")\n\n")
    print(x$summary, row.names = FALSE)
    cat("\nGlobal minimum: f =", min(x$summary$f), "\n")
    cat("Access full solutions via $minima[[i]]$result\n")
    invisible(x)
  }
```

## Key Arguments

- `method` — the rotation criterion. Criteria with many local minima, such as `simplimax`, `geomin`, and `infomax`, are the most interesting to diagnose.
- `randomStarts` — number of random starts to attempt. More starts give a more complete picture of the solution space. For a thorough analysis, 500 or more starts are recommended.
- `minimumInclusion` — minimum number of starts required to retain a minimum. The default of 2 filters out singletons that are likely numerical artifacts. With 500 starts, a value of 5 or 10 is more appropriate.

- `orthogonal` — `TRUE` for orthogonal rotation (uses `GPForth`), `FALSE` for oblique (uses `GPFoblq`). Default is `FALSE`.

Non-converged starts are always discarded before analysis. The `proportion` column in the summary is calculated over converged starts only, so proportions sum to 1.

## Example: Simplimax on CCAI Climate-Friendly Purchasing Choices Data

The CCAI Climate-Friendly Purchasing Choices domain (Bi and Barchard, 2024) provides a useful dataset for illustrating local minima behavior. The data have 14 items and 3 factors with strong inter-correlations (0.53–0.59). Bi and Barchard used oblimin rotation in their published analysis. We use simplimax here purely to illustrate how rotation criteria can produce highly complex landscapes with multiple local minima — not as a recommended analysis of these data.

The data illustrate how dramatically rotation criteria can differ in their stability. Oblimin rotation is highly stable on these data — all 200 random starts converge to the same solution. Simplimax, by contrast, reveals a highly complex landscape:

```
> library("GPArotation")
> data("CCAI", package = "GPArotation")
> fa_unrotated <- factanal(factors = 3, covmat = CCAI_R,
        n.obs = 461, rotation = "none")
> A <- loadings(fa_unrotated)
> # Oblimin: highly stable
> res_oblimin <- oblimin(A, normalize = TRUE, randomStarts = 200)
> cat("Oblimin random start diagnostics:\n")

Oblimin random start diagnostics:

> res_oblimin$randStartChar

randomStarts    Converged    atMinimum    localMins
         200          200          200            1

> # Simplimax: complex landscape
> set.seed(42)
> res_ccai <- GPFallMinima(A, method = "simplimax",
                           randomStarts = 200,
                           normalize = TRUE,
                           minimumInclusion = 2)
> res_ccai

Random start analysis: 65 of 200 starts converged
Distinct minima found: 13 (minimumInclusion = 2 )

 minimum       f  deltaF count proportion isGlobal
       1 0.06125 0.05045    10      0.154    FALSE
```

```
 2 0.07500 0.06420     7      0.108    FALSE
 3 0.21134 0.20054     7      0.108    FALSE
 4 0.05224 0.04144     6      0.092    FALSE
 5 0.03311 0.02231     6      0.092    FALSE
 6 0.09336 0.08256     4      0.062    FALSE
 7 0.03421 0.02341     3      0.046    FALSE
 8 0.08631 0.07551     3      0.046    FALSE
 9 0.05323 0.04243     3      0.046    FALSE
10 0.07323 0.06243     3      0.046    FALSE
11 0.09342 0.08262     2      0.031    FALSE
12 0.02875 0.01795     2      0.031    FALSE
13 0.01080 0.00000     2      0.031     TRUE

Global minimum: f = 0.0108
Access full solutions via $minima[[i]]$result
```

The contrast is striking. Oblimin converges reliably to a single solution on these data. Simplimax reveals 16 distinct local minima, with only 81 of 200 starts converging (40.5%) and the global minimum found in just 2.5% of converged starts. This underscores the importance of using multiple random starts and verifying convergence, particularly when using criteria known to be prone to local minima such as simplimax.

The global minimum (f = 0.01080) is clearly separated from the other local minima, with the next best solution having a criterion value three times larger. The sorted loadings plot shows that the local minima produce substantially different factor structures, not merely permutations or sign flips of the same solution.

## Examining Individual Solutions

Each element of `res_ccai$minima` contains a full `GPArotation` object in `result`, so the standard `print` and `summary` methods work directly.

```
> # Print the most common solution
> print(res_ccai$minima[[1]]$result)

Oblique rotation method Simplimax (k=14) converged.
Loadings:
       Factor1 Factor2 Factor3
CCAI8    0.929   0.001  -0.099
CCAI6    0.852   0.008  -0.053
CCAI7    0.836  -0.041   0.020
CCAI11   0.832   0.196   0.097
CCAI12   0.803   0.273   0.060
CCAI10   0.745   0.217   0.165
CCAI14   0.596   0.780  -0.040
CCAI13   0.617   0.738  -0.005
CCAI5    0.613   0.491   0.189
```

```
CCAI2    0.458  -0.065   0.439
CCAI4    0.593   0.033   0.560
CCAI1    0.552   0.001   0.489
CCAI3    0.644   0.085   0.436
CCAI9    0.678   0.137   0.344

                Factor1 Factor2 Factor3
SS loadings       7.033   1.654   1.212
Proportion Var    0.502   0.118   0.087
Cumulative Var    0.502   0.621   0.707

Phi:
        Factor1 Factor2 Factor3
Factor1       1   0.000   0.000
Factor2       0   1.000   0.328
Factor3       0   0.328   1.000
```

Accessing any specific solution is straightforward. For example, to print the solution corresponding to row 3 of the summary table:

```
> # Print solution in row 3 of the summary
> print(res_ccai$minima[[3]]$result, digits = 2)

Oblique rotation method Simplimax (k=14) converged.
Loadings:
       Factor1 Factor2 Factor3
CCAI8     0.66    0.52    0.43
CCAI6     0.63    0.45    0.38
CCAI7     0.65    0.38    0.38
CCAI11    0.76    0.41    0.13
CCAI12    0.73    0.45    0.07
CCAI10    0.74    0.33    0.04
CCAI14    0.63    0.58   -0.43
CCAI13    0.66    0.56   -0.40
CCAI5     0.72    0.34   -0.27
CCAI2     0.62   -0.08    0.05
CCAI4     0.82   -0.06   -0.04
CCAI1     0.74   -0.05    0.00
CCAI3     0.80    0.06   -0.01
CCAI9     0.78    0.16    0.00

                Factor1 Factor2 Factor3
SS loadings        7.09    1.90    0.91
Proportion Var     0.51    0.14    0.06
Cumulative Var     0.51    0.64    0.71

Phi:
```

```
        Factor1 Factor2 Factor3
Factor1       1    0.00    0.00
Factor2       0    1.00   -0.05
Factor3       0   -0.05    1.00
```

More generally, to print the solution in row `i`:

```
> i <- 3
> print(res_ccai$minima[[i]]$result, digits = 2)

Oblique rotation method Simplimax (k=14) converged.
Loadings:
       Factor1 Factor2 Factor3
CCAI8     0.66    0.52    0.43
CCAI6     0.63    0.45    0.38
CCAI7     0.65    0.38    0.38
CCAI11    0.76    0.41    0.13
CCAI12    0.73    0.45    0.07
CCAI10    0.74    0.33    0.04
CCAI14    0.63    0.58   -0.43
CCAI13    0.66    0.56   -0.40
CCAI5     0.72    0.34   -0.27
CCAI2     0.62   -0.08    0.05
CCAI4     0.82   -0.06   -0.04
CCAI1     0.74   -0.05    0.00
CCAI3     0.80    0.06   -0.01
CCAI9     0.78    0.16    0.00

               Factor1 Factor2 Factor3
SS loadings       7.09    1.90    0.91
Proportion Var    0.51    0.14    0.06
Cumulative Var    0.51    0.64    0.71

Phi:
        Factor1 Factor2 Factor3
Factor1       1    0.00    0.00
Factor2       0    1.00   -0.05
Factor3       0   -0.05    1.00
```

The row number in the summary table corresponds directly to the index in `res_ccai$minima`.
Row 1 is always the most common solution, and the global minimum is identified by `res_ccai$summary$isGlobal`.

```
> # Print the global minimum using the GPArotation S3 print method
> global <- which(res_ccai$summary$isGlobal)
> print(res_ccai$minima[[global]]$result, digits = 2)

Oblique rotation method Simplimax (k=14) converged.
Loadings:
```

```
       Factor1 Factor2 Factor3
CCAI8     0.94   -0.03    0.02
CCAI6     0.82    0.03    0.02
CCAI7     0.77    0.13   -0.04
CCAI11    0.51    0.22    0.25
CCAI12    0.47    0.16    0.35
CCAI10    0.35    0.30    0.28
CCAI14   -0.01   -0.02    0.99
CCAI13    0.01    0.03    0.94
CCAI5     0.00    0.30    0.62
CCAI2     0.04    0.65   -0.10
CCAI4    -0.03    0.83    0.02
CCAI1     0.03    0.73   -0.01
CCAI3     0.10    0.66    0.10
CCAI9     0.18    0.54    0.17

               Factor1 Factor2 Factor3
SS loadings       3.45    3.33    3.12
Proportion Var    0.25    0.24    0.22
Cumulative Var    0.25    0.48    0.71

Phi:
        Factor1 Factor2 Factor3
Factor1    1.00    0.67    0.59
Factor2    0.67    1.00    0.63
Factor3    0.59    0.63    1.00

> # Summary with structure matrix for oblique solution
> summary(res_ccai$minima[[global]]$result, Structure = TRUE, digits = 2)

Oblique rotation method Simplimax (k=14) converged in 96 iterations.
Pattern (loadings):
       Factor1 Factor2 Factor3
CCAI8     0.94   -0.03    0.02
CCAI6     0.82    0.03    0.02
CCAI7     0.77    0.13   -0.04
CCAI11    0.51    0.22    0.25
CCAI12    0.47    0.16    0.35
CCAI10    0.35    0.30    0.28
CCAI14   -0.01   -0.02    0.99
CCAI13    0.01    0.03    0.94
CCAI5     0.00    0.30    0.62
CCAI2     0.04    0.65   -0.10
CCAI4    -0.03    0.83    0.02
CCAI1     0.03    0.73   -0.01
CCAI3     0.10    0.66    0.10
CCAI9     0.18    0.54    0.17
```

```
Structure:
       Factor1 Factor2 Factor3
CCAI8     0.93    0.61    0.56
CCAI6     0.85    0.59    0.53
CCAI7     0.83    0.62    0.50
CCAI11    0.81    0.72    0.70
CCAI12    0.78    0.69    0.73
CCAI10    0.72    0.71    0.67
CCAI14    0.56    0.60    0.97
CCAI13    0.58    0.63    0.96
CCAI5     0.57    0.70    0.81
CCAI2     0.42    0.62    0.34
CCAI4     0.54    0.82    0.53
CCAI1     0.51    0.74    0.46
CCAI3     0.60    0.79    0.57
CCAI9     0.64    0.76    0.61
```

Solutions with small `deltaF` values may produce loading matrices that are very similar to the global minimum after sorting. Solutions with large `deltaF` values are likely to produce meaningfully different loading patterns and warrant careful examination.

## Visualizing All Minima

The sorted absolute loadings plot is a powerful tool for comparing all retained minima at once. Each line represents one distinct solution. Lines that overlap closely indicate practically equivalent solutions; lines that diverge indicate qualitatively different solutions.

The following plotting function is also used in the main `GPA1guide` vignette. It is reproduced here so that this vignette is self-contained.

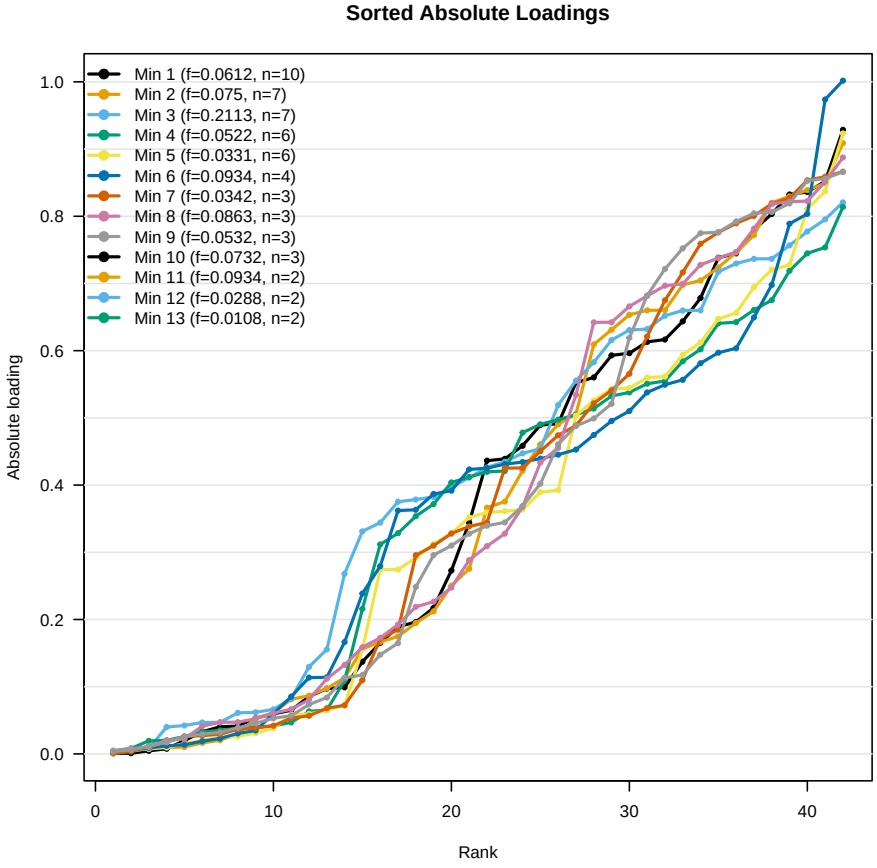
First define the plotting function:

```
> 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)
}
```

Then plot all retained minima:

```
> do.call(plotSortedLoadings,
          c(lapply(res_ccai$minima, function(x) x$result),
            list(labels = paste0("Min ", res_ccai$summary$minimum,
                                 " (f=", round(res_ccai$summary$f, 4),
                                 ", n=", res_ccai$summary$count, ")"))))
```




Lines that are visually indistinguishable correspond to solutions that are practically equivalent regardless of their `deltaF`. Lines that diverge clearly represent genuinely different factor structures that merit substantive interpretation.

# Practical Guidance

- **How many random starts?** For criteria that tend to have many local minima, 500 or more starts are recommended. The goal is for the proportions in the summary to stabilize — if running 1000 instead of 500 starts changes the proportions substantially, more starts are needed.
- **Setting minimumInclusion.** With 100 starts, a value of 2 is reasonable. With 500 starts, use 5 or 10. The goal is to distinguish genuine local minima from numerical noise.
- **Interpreting deltaF.** There is no universal threshold for what constitutes a "meaningful" difference in criterion value. Solutions with `deltaF` $< 0.001$ are typically negligible. Solutions with `deltaF` $> 0.01$ may produce visibly different loading patterns.

- **When the global minimum has low proportion.** If the global minimum is found by fewer than 20% of converged starts, the criterion landscape is complex and the solution may not be stable. Consider trying a different rotation criterion or increasing the number of random starts.
- **Comparing criteria.** Running `GPFallMinima` with different rotation criteria on the same dataset can reveal which criteria produce stable solutions and which are prone to local minima for a given data structure.

## Further Resources

For detailed discussion of local minima in factor rotation and their implications for substantive interpretation, see Nguyen & Waller (2022). For investigation of local minima using the *fungible* package, see the `faMain` function. The *psych* package provides `faRotations` for similar diagnostics.

The `randStartChar` element returned by standard `GPArotation` functions provides a quick summary of random start results without retaining individual solutions. For routine use, `randStartChar` is sufficient; `GPFallMinima` is intended for deeper diagnostic investigation.

## References


Bi, Y. and Barchard, K.A. (2024). Purchasing choices that reduce climate change: An exploratory factor analysis. *Spectra Undergraduate Research Journal*, **3**(2), 8–14. doi: 10.9741/2766-7227.1028

Mansolf, M., & Reise, S. P. (2016). Exploratory bifactor analysis: The Schmid-Leiman orthogonalization and Jennrich-Bentler analytic rotations. *Multivariate Behavioral Research*, 51(5), 698–717. https://doi.org/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. https://doi.org/10.1037/met0000467