

# Package ‘kfa’

May 8, 2026

**Type** Package

**Title** K-Fold Cross Validation for Factor Analysis

**Version** 0.2.2

**Author** Kyle Nickodem [aut, cre] and Peter Halpin [aut]

**Maintainer** Kyle Nickodem <kyle.nickodem@gmail.com>

**Description** Provides functions to identify plausible and replicable factor structures for a set of variables via k-fold cross validation. The process combines the exploratory and confirmatory factor analytic approach to scale development (Flora & Flake, 2017) <doi:10.1037/cbs000069> with a cross validation technique that maximizes the available data (Hastie, Tibshirani, & Friedman, 2009) <isbn:978-0-387-21606-5>. Also available are functions to determine k by drawing on power analytic techniques for covariance structures (MacCallum, Browne, & Sugawara, 1996) <doi:10.1037/1082-989X.1.2.130>, generate model syntax, and summarize results in a report.

**Depends** R (>= 3.6)

**Imports** caret, doParallel, flextable (>= 0.6.3), foreach, GPArotation, knitr, lavaan (>= 0.6.9), officer, parallel, rmarkdown, semTools (>= 0.5.5), simstandard

**Suggests** semPlot

**License** GPL (>= 3)

**Encoding** UTF-8

**LazyData** true

**URL** <https://github.com/knickodem/kfa>

**BugReports** <https://github.com/knickodem/kfa/issues>

**RoxygenNote** 7.2.3

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2023-07-09 09:00:02 UTC

## Contents

agg_cors . . . . .	2
agg_loadings . . . . .	3
agg_model_fit . . . . .	3
agg_rels . . . . .	4
efa_cfa_syntax . . . . .	5
example.kfa . . . . .	6
find_k . . . . .	7
get_std_loadings . . . . .	8
index_available . . . . .	9
kfa . . . . .	9
kfa_report . . . . .	12
k_model_fit . . . . .	13
model_structure . . . . .	14
run_efa . . . . .	15
write_efa . . . . .	17
<b>Index</b>	<b>18</b>

---

agg_cors	<i>Aggregated factor correlations</i>
----------	---------------------------------------

---

### Description

The factor correlations aggregated over k-folds

### Usage

```
agg_cors(models, flag = 0.9, type = "factor")
```

### Arguments

models	An object returned from <a href="#">kfa</a>
flag	threshold above which a factor correlation will be flagged
type	currently ignored; "factor" (default) or "observed" variable correlations

### Value

data.frame of mean factor correlations for each factor model and vector with count of folds with a flagged correlation

### Examples

```
data(example.kfa)
agg_cors(example.kfa)
```

---

agg_loadings	<i>Aggregated factor loadings</i>
--------------	-----------------------------------

---

**Description**

The factor loadings aggregated over k-folds

**Usage**

```
agg_loadings(models, flag = 0.3, digits = 2)
```

**Arguments**

models	An object returned from <a href="#">kfa</a>
flag	threshold below which loading will be flagged
digits	integer; number of decimal places to display in the report.

**Value**

data.frame of mean factor loadings for each factor model and vector with count of folds with a flagged loading

**Examples**

```
data(example.kfa)
agg_loadings(example.kfa)
```

---

agg_model_fit	<i>Summary table of model fit</i>
---------------	-----------------------------------

---

**Description**

Summary table of model fit aggregated over k-folds

**Usage**

```
agg_model_fit(kfits, index = "all", digits = 2)
```

**Arguments**

kfits	an object returned from <a href="#">k_model_fit</a> when <code>by.folds = TRUE</code>
index	character; one or more fit indices to summarize. Indices must be present in the kfits object. Default is "all" indices present in kfits. Chi-square value and degrees of freedom are always reported.
digits	integer; number of decimal places to display in the report

**Value**

data.frame of aggregated model fit statistics

**Examples**

```
data(example.kfa)
fits <- k_model_fit(example.kfa, by.fold = TRUE)
agg_model_fit(fits)
```

---

agg_rels	<i>Aggregated scale reliabilities</i>
----------	---------------------------------------

---

**Description**

The factor reliabilities aggregated over k-folds

**Usage**

```
agg_rels(models, flag = 0.6, digits = 2)
```

**Arguments**

models	An object returned from <a href="#">kfa</a>
flag	threshold below which reliability will be flagged
digits	integer; number of decimal places to display in the report.

**Value**

data.frame of mean factor (scale) reliabilities for each factor model and vector with count of folds with a flagged reliability

**Examples**

```
data(example.kfa)
agg_rels(example.kfa)
```

---

efa_cfa_syntax	<i>Write confirmatory factor analysis syntax</i>
----------------	--

---

### Description

Uses the factor loadings matrix, presumably from an exploratory factor analysis, to generate lavaan compatible confirmatory factory analysis syntax.

### Usage

```
efa_cfa_syntax(
  loadings,
  simple = TRUE,
  min.loading = NA,
  single.item = c("keep", "drop", "none"),
  identified = TRUE,
  constrain0 = FALSE
)
```

### Arguments

loadings	matrix of factor loadings
simple	logical; Should the perfect simple structure be returned (default) when converting EFA results to CFA syntax? If FALSE, items can cross-load on multiple factors.
min.loading	numeric between 0 and 1 indicating the minimum (absolute) value of the loading for a variable on a factor when converting EFA results to CFA syntax. Must be specified when simple = FALSE.
single.item	character indicating how single-item factors should be treated. Use "keep" (default) to keep them in the model when generating the CFA syntax, "drop" to remove them, or "none" indicating the CFA syntax should not be generated for this model and "" is returned.
identified	logical; Should identification check for rotational uniqueness a la Millsap (2001) be performed? If the model is not identified "" is returned.
constrain0	logical; Should variable(s) with all loadings below min.loading still be included in model syntax? If TRUE, variable(s) will load onto first factor with the loading constrained to 0.

### References

Millsap, R. E. (2001). When trivial constraints are not trivial: The choice of uniqueness constraints in confirmatory factor analysis. *Structural Equation Modeling*, 8\*(1), 1-17. doi:10.1207/S15328007SEM0801\_1

## Examples

```
loadings <- matrix(c(rep(.2, 3), rep(.6, 3), rep(.8, 3), rep(.3, 3)), ncol = 2)
# simple structure
efa_cfa_syntax(loadings)
# allow cross-loadings and check if model is identified
efa_cfa_syntax(loadings, simple = FALSE, min.loading = .25)
# allow cross-loadings and ignore identification check
efa_cfa_syntax(loadings, simple = FALSE, min.loading = .25, identified = FALSE)
```

---

example.kfa

*kfa results from simulated data example*

---

## Description

Simulated responses for 900 observations on 20 variables loading onto a 3 factor structure (see example in [kfa](#) documentation for model). The simulated data was run through [kfa](#) with the call `kfa(sim.data, k = 2, m = 3)` which tested 1-, 2-, and 3-factor structures over 2 folds.

## Usage

```
data(example.kfa)
```

## Format

An object of class "kfa", which is a four-element list:

- **cfas** lavaan CFA objects for each  $k$  fold
- **cfa.syntax** syntax used to produce CFA objects
- **model.names** vector of names for CFA objects
- **efa.structures** all factor structures identified in the EFA

## Examples

```
data(example.kfa)
agg_cors(example.kfa)
```

---

find_k	<i>Find k for k-fold cross-validation</i>
--------	---

---

### Description

This function is specifically for determining  $k$  in the context of factor analysis using change in RMSEA as the criterion for identifying the optimal factor model.

### Usage

```
find_k(
  variables,
  n,
  p,
  m = NULL,
  max.k = 10,
  min.n = 200,
  rmsea0 = 0.05,
  rmseaA = 0.08,
  ...
)
```

### Arguments

variables	a data.frame (or convertible to a data.frame) with variables to factor analyze in columns and observations in rows. The power analysis assumes all observations have complete data. Use n argument or remove rows manually to account for missingness.
n	integer; number of observations. Ignored if variables is provided.
p	integer; number of variables to factor analyze. Ignored if variables is provided.
m	integer; maximum number of factors expected to be extracted from variables. Default is $p / 4$ (i.e., 4 variables per factor).
max.k	integer; maximum number of folds. Default is 10. NULL indicates no maximum.
min.n	integer; minimum sample size per fold. Default is 200 based on simulations from Curran et al. (2003).
rmsea0	numeric; RMSEA under the null hypothesis.
rmseaA	numeric; RMSEA under the alternative hypothesis.
...	other arguments passed to <a href="#">findRMSEAsamplesize</a> .

### Value

named vector with the number of folds (k), sample size suggested by the power analysis (power.n), and the actual sample size used for determining k (actual.n).

## References

Curran, P. J., Bollen, K. A., Chen, F., Paxton, P., & Kirby, J. B. (2003). Finite sampling properties of the point estimates and confidence intervals of the RMSEA. *Sociological Methods & Research*, 32(2), 208-252. doi:10.1177/0049124103256130

MacCallum, R. C., Browne, M. W., & Sugawara, H. M. (1996). Power analysis and determination of sample size for covariance structure modeling. *Psychological Methods*, 1(2), 130-149. doi:10.1037/1082989X.1.2.130

## Examples

```
find_k(n = 900, p = 20, m = 3)

# adjust precision
find_k(n = 900, p = 20, m = 3, rmsea0 = .03, rmseaA = .10)
```

---

<code>get_std_loadings</code>	<i>Standardized factor loadings matrix</i>
-------------------------------	--

---

## Description

Extract standardized factor loadings from lavaan object

## Usage

```
get_std_loadings(object, type = "std.all", df = FALSE)
```

## Arguments

<code>object</code>	a lavaan object
<code>type</code>	standardize on the latent variables ("std.lv"), latent and observed variables ("std.all", default), or latent and observed variables but not exogenous variables ("std.nox")? See <a href="#">standardizedSolution</a> .
<code>df</code>	should loadings be returned as a matrix (default) or data.frame?

## Value

A matrix or data.frame of factor loadings

## Examples

```
data(HolzingerSwineford1939, package = "lavaan")
HS.model <- ' visual  =~ x1 + x2 + x3
              textual =~ x4 + x5 + x6
              speed   =~ x7 + x8 + x9 '
```

```
fit <- lavaan::cfa(HS.model, data = HolzingerSwineford1939)
get_std_loadings(fit)
```

---

index_available	<i>Available Fit Indices</i>
-----------------	------------------------------

---

**Description**

Shows the fit indices available from `kfa` object to report in `kfa_report`

**Usage**

```
index_available(models)
```

**Arguments**

`models` an object returned from `kfa`

**Value**

character vector of index names

**Examples**

```
data(example.kfa)
index_available(example.kfa)
```

---

<code>kfa</code>	<i>Conducts k-fold cross validation for factor analysis</i>
------------------	---

---

**Description**

The function splits the data into  $k$  folds where each fold contains training data and test data. For each fold, exploratory factor analyses (EFAs) are run on the training data. The structure for each model is transformed into lavaan-compatible confirmatory factor analysis (CFA) syntax. The CFAs are then run on the test data.

**Usage**

```
kfa(
  data,
  variables = names(data),
  k = NULL,
  m = floor(length(variables)/4),
  seed = 101,
  cores = NULL,
  custom.cfas = NULL,
  power.args = list(rmseath = 0.05, rmseaA = 0.08),
```

```

rotation = "oblimin",
simple = TRUE,
min.loading = NA,
single.item = "none",
ordered = FALSE,
estimator = NULL,
missing = "listwise",
...
)

```

## Arguments

<code>data</code>	a <code>data.frame</code> containing the variables (i.e., items) to factor analyze
<code>variables</code>	character vector of column names in <code>data</code> indicating the variables to factor analyze. Default is to use all columns.
<code>k</code>	number of folds in which to split the data. Default is <code>NULL</code> which determines <code>k</code> via <a href="#">find_k</a> .
<code>m</code>	integer; maximum number of factors to extract. Default is 4 items per factor.
<code>seed</code>	integer passed to <code>set.seed</code> when randomly selecting cases for each fold.
<code>cores</code>	integer; number of CPU cores to use for parallel processing. Default is <code>detectCores - 1</code> .
<code>custom.cfas</code>	a single object or named list of lavaan syntax specifying custom factor model(s).
<code>power.args</code>	named list of arguments to pass to <a href="#">find_k</a> and <a href="#">findRMSEAsamplesize</a> when conducting power analysis to determine <code>k</code> .
<code>rotation</code>	character (case-sensitive); any rotation method listed in <a href="#">rotations</a> in the <code>GPARotation</code> package. Default is "oblimin".
<code>simple</code>	logical; Should the perfect simple structure be returned (default) when converting EFA results to CFA syntax? If <code>FALSE</code> , items can cross-load on multiple factors.
<code>min.loading</code>	numeric between 0 and 1 indicating the minimum (absolute) value of the loading for a variable on a factor when converting EFA results to CFA syntax. Must be specified when <code>simple = FALSE</code> .
<code>single.item</code>	character indicating how single-item factors should be treated. Use "keep" to keep them in the model when generating the CFA syntax or "none" (default) indicating the CFA syntax should not be generated for this model and "" is returned.
<code>ordered</code>	logical; Should items be treated as ordinal and the polychoric correlations used in the factor analysis? When <code>FALSE</code> (default) the Pearson correlation matrix is used. A character vector of item names is also accepted to prompt estimation of the polychoric correlation matrix.
<code>estimator</code>	if <code>ordered = FALSE</code> , the default is "MLMVS". If <code>ordered = TRUE</code> , the default is "WLSMV". See <a href="#">lavOptions</a> for other options.
<code>missing</code>	default is "listwise". See <a href="#">lavOptions</a> for other options.
<code>...</code>	other arguments passed to lavaan functions. See <a href="#">lavOptions</a> .

## Details

In order for `custom.cfas` to be tested along with the EFA identified structures, each model supplied in `custom.cfas` must include all variables in lavaan-compatible syntax.

Deciding an appropriate  $m$  can be difficult, but is consequential for the possible factor structures to examine, the power analysis to determine  $k$ , and overall computation time. The `n_factors` function in the `parameters` package can assist with this decision.

When converting EFA results to CFA syntax (via `efa_cfa_syntax`), the simple structure is defined as each variable loading onto a single factor. This is determined using the largest factor loading for each variable. When `simple = FALSE`, variables are allowed to cross-load on multiple factors. In this case, all pathways with loadings above the `min.loading` are retained. However, allowing cross-loading variables can result in model under-identification. The `efa_cfa_syntax` function conducts an identification check (i.e., `identified = TRUE`) and under-identified models are not run in the CFA portion of the analysis.

## Value

An object of class "kfa", which is a four-element list:

- `cfas` lavaan CFA objects for each  $k$  fold
- `cfa.syntax` syntax used to produce CFA objects
- `model.names` vector of names for CFA objects
- `efa.structures` all factor structures identified in the EFA

## Examples

```
# simulate data based on a 3-factor model with standardized loadings
sim.mod <- "f1 =~ .7*x1 + .8*x2 + .3*x3 + .7*x4 + .6*x5 + .8*x6 + .4*x7
          f2 =~ .8*x8 + .7*x9 + .6*x10 + .5*x11 + .5*x12 + .7*x13 + .6*x14
          f3 =~ .6*x15 + .5*x16 + .9*x17 + .4*x18 + .7*x19 + .5*x20
          f1 ~~ .2*f2
          f2 ~~ .2*f3
          f1 ~~ .2*f3
          x9 ~~ .2*x10"

set.seed(1161)
sim.data <- simstandard::sim_standardized(sim.mod, n = 900,
                                         latent = FALSE,
                                         errors = FALSE)[c(2:9,1,10:20)]

# include a custom 2-factor model
custom2f <- paste0("f1 =~ ", paste(colnames(sim.data)[1:10], collapse = " + "),
                  "\nf2 =~ ", paste(colnames(sim.data)[11:20], collapse = " + "))

mods <- kfa(data = sim.data,
            k = NULL, # prompts power analysis to determine number of folds
            cores = 2,
            custom.cfas = custom2f)
```

---

kfa_report	<i>Creates summary report from a k-fold factor analysis</i>
------------	---

---

### Description

Generates a report summarizing the factor analytic results over k-folds.

### Usage

```
kfa_report(
  models,
  file.name,
  report.title = file.name,
  path = NULL,
  report.format = "html_document",
  word.template = NULL,
  index = "default",
  plots = FALSE,
  load.flag = 0.3,
  cor.flag = 0.9,
  rel.flag = 0.6,
  digits = 2
)
```

### Arguments

models	an object returned from <a href="#">kfa</a>
file.name	character; file name to create on disk.
report.title	character; title of the report
path	character; path of the directory where summary report will be saved. Default is working directory. path and file.name are combined to create final file path
report.format	character; file format of the report. Default is HTML ("html_document"). See <a href="#">render</a> for other options.
word.template	character; file path to word document to use as a formatting template when report.format = "word_document".
index	character; one or more fit indices to summarize in the report. Use <a href="#">index_available</a> to see choices. Chi-square value and degrees of freedom are always reported. Default is CFI and RMSEA (naive, scaled, or robust version depends on estimator used in models).
plots	logical; should plots of the factor models be included in the report?
load.flag	numeric; factor loadings of variables below this value will be flagged. Default is .30
cor.flag	numeric; factor correlations above this value will be flagged. Default is .90
rel.flag	numeric; factor (scale) reliabilities below this value will be flagged. Default is .60.
digits	integer; number of decimal places to display in the report.

**Value**

A summary report of factor structures and model fit within and between folds.

**Examples**

```
# simulate data based on a 3-factor model with standardized loadings
sim.mod <- "f1 =~ .7*x1 + .8*x2 + .3*x3 + .7*x4 + .6*x5 + .8*x6 + .4*x7
          f2 =~ .8*x8 + .7*x9 + .6*x10 + .5*x11 + .5*x12 + .7*x13 + .6*x14
          f3 =~ .6*x15 + .5*x16 + .9*x17 + .4*x18 + .7*x19 + .5*x20
          f1 ~~ .2*f2
          f2 ~~ .2*f3
          f1 ~~ .2*f3
          x9 ~~ .2*x10"

set.seed(1161)
sim.data <- simstandard::sim_standardized(sim.mod, n = 900,
                                          latent = FALSE,
                                          errors = FALSE)[c(2:9,11,10:20)]

# include a custom 2-factor model
custom2f <- paste0("f1 =~ ", paste(colnames(sim.data)[1:10], collapse = " + "),
                  "\nf2 =~ ", paste(colnames(sim.data)[11:20], collapse = " + "))

mods <- kfa(data = sim.data,
            k = NULL, # prompts power analysis to determine number of folds
            cores = 2,
            custom.cfas = custom2f)

## Not run:
kfa_report(mods, file.name = "example_sim_kfa_report",
           report.format = "html_document",
           report.title = "K-fold Factor Analysis - Example Sim")

## End(Not run)
```

---

k\_model\_fit

*Extract model fit*


---

**Description**

Model fit indices extracted from k-folds

**Usage**

```
k_model_fit(models, index = "default", by.fold = TRUE)
```

**Arguments**

models	an object returned from <a href="#">kfa</a>
index	character; one or more fit indices to summarize in the report. Use <a href="#">index_available</a> to see choices. Chi-square value and degrees of freedom are always reported. Default is CFI and RMSEA (naive, scaled, or robust version depends on estimator used in models).
by.fold	Should each element in the returned lists be a fold (default) or a factor model?

**Value**

list of data.frames with average model fit for each factor model

**Examples**

```
data(example.kfa)

# customize fit indices to report
k_model_fit(example.kfa, index = c("chisq", "cfi", "rmsea", "srmr"))

# organize results by factor model rather than by fold
k_model_fit(example.kfa, by.fold = FALSE)
```

---

model_structure	<i>Unique factor structures</i>
-----------------	---------------------------------

---

**Description**

Extract unique factor structures across the k-folds

**Usage**

```
model_structure(models)
```

**Arguments**

models	An object returned from <a href="#">kfa</a>
--------	---

**Value**

data.frame with the number of folds the unique factor structure was tested for each factor model.

**Examples**

```
data(example.kfa)
model_structure(example.kfa)
```

---

run_efa	<i>Conducts exploratory factor analysis</i>
---------	---

---

### Description

This function is intended for use on independent samples rather than integrated with k-fold cross-validation.

### Usage

```
run_efa(
  data,
  variables = names(data),
  m = floor(ncol(data)/4),
  rotation = "oblimin",
  simple = TRUE,
  min.loading = NA,
  single.item = c("keep", "drop", "none"),
  identified = TRUE,
  constrain0 = FALSE,
  ordered = FALSE,
  estimator = NULL,
  missing = "listwise",
  ...
)
```

### Arguments

<code>data</code>	a data.frame containing the variables (i.e., items) to factor analyze
<code>variables</code>	character vector of column names in data indicating the variables to factor analyze. Default is to use all columns.
<code>m</code>	integer; maximum number of factors to extract. Default is 4 items per factor.
<code>rotation</code>	character (case-sensitive); any rotation method listed in <a href="#">rotations</a> in the GPArotation package. Default is "oblimin".
<code>simple</code>	logical; Should the perfect simple structure be returned (default) when converting EFA results to CFA syntax? If FALSE, items can cross-load on multiple factors.
<code>min.loading</code>	numeric between 0 and 1 indicating the minimum (absolute) value of the loading for a variable on a factor when converting EFA results to CFA syntax. Must be specified when <code>simple = FALSE</code> .
<code>single.item</code>	character indicating how single-item factors should be treated. Use "keep" (default) to keep them in the model when generating the CFA syntax, "drop" to remove them, or "none" indicating the CFA syntax should not be generated for this model and "" is returned.

identified	logical; Should identification check for rotational uniqueness a la Millsap (2001) be performed? If the model is not identified "" is returned.
constrain0	logical; Should variable(s) with all loadings below min.loading still be included in model syntax? If TRUE, variable(s) will load onto first factor with the loading constrained to 0.
ordered	logical; Should items be treated as ordinal and the polychoric correlations used in the factor analysis? When FALSE (default) the Pearson correlation matrix is used. A character vector of item names is also accepted to prompt estimation of the polychoric correlation matrix.
estimator	if ordered = FALSE, the default is "MLMVS". If ordered = TRUE, the default is "WLSMV". See <a href="#">lavOptions</a> for other options.
missing	default is "listwise". See <a href="#">lavOptions</a> for other options.
...	other arguments passed to lavaan functions. See <a href="#">lavOptions</a> .

### Details

When converting EFA results to CFA syntax (via [efa\\_cfa\\_syntax](#)), the simple structure is defined as each variable loading onto a single factor. This is determined using the largest factor loading for each variable. When `simple = FALSE`, variables are allowed to cross-load on multiple factors. In this case, all pathways with loadings above the `min.loading` are retained. However, allowing cross-loading variables can result in model under-identification. An identification check is run by default, but can be turned off by setting `identified = FALSE`.

### Value

A three-element list:

- **efas** lavaan object for each *m* model
- **loadings** (rotated) factor loading matrix for each *m* model
- **cfa.syntax** CFA syntax generated from loadings

### References

Millsap, R. E. (2001). When trivial constraints are not trivial: The choice of uniqueness constraints in confirmatory factor analysis. *Structural Equation Modeling*, 8(1), 1-17. doi:10.1207/S15328007SEM0801\_1

### Examples

```
# simulate data based on a 3-factor model with standardized loadings
sim.mod <- "f1 =~ .7*x1 + .8*x2 + .3*x3 + .7*x4 + .6*x5 + .8*x6 + .4*x7
           f2 =~ .8*x8 + .7*x9 + .6*x10 + .5*x11 + .5*x12 + .7*x13 + .6*x14
           f3 =~ .6*x15 + .5*x16 + .9*x17 + .4*x18 + .7*x19 + .5*x20
           f1 ~~ .2*f2
           f2 ~~ .2*f3
           f1 ~~ .2*f3
           x9 ~~ .2*x10"

set.seed(1161)
```

```
sim.data <- simstandard::sim_standardized(sim.mod, n = 900,  
                                          latent = FALSE,  
                                          errors = FALSE)[c(2:9,1,10:20)]  
  
# Run 1-, 2-, and 3-factor models  
efas <- run_efa(sim.data, m = 3)
```

---

write\_efa

*Write exploratory factor analysis syntax*

---

### Description

Converts variable names to lavaan-compatible exploratory factor analysis syntax

### Usage

```
write_efa(nf, vnames)
```

### Arguments

nf	integer; number of factors
vnames	character vector; names of variables to include in the efa

### Value

character. Use `cat()` to best examine the returned syntax.

### Examples

```
vnames <- paste("x", 1:10)  
syntax <- write_efa(nf = 2, vnames = vnames)  
cat(syntax)
```

# Index

## \* datasets

example.kfa, 6

agg\_cors, 2

agg\_loadings, 3

agg\_model\_fit, 3

agg\_rels, 4

detectCores, 10

efa\_cfa\_syntax, 5, 11, 16

example.kfa, 6

find\_k, 7, 10

findRMSEAsamplesize, 7, 10

get\_std\_loadings, 8

index\_available, 9, 12, 14

k\_model\_fit, 3, 13

kfa, 2–4, 6, 9, 9, 12, 14

kfa\_report, 9, 12

lavOptions, 10, 16

model\_structure, 14

render, 12

rotations, 10, 15

run\_efa, 15

standardizedSolution, 8

write\_efa, 17