

# Package: ipf (via r-universe)

June 8, 2026

**Title** Iterative Proportional Fitting

**Version** 0.0.0.9000

**Description** Fast raking for survey weighting. The computational core is written in Rust for speed. Supports multiple raking variables, automatic variable selection, weight bounding, and comprehensive diagnostics.

**License** MIT + file LICENSE

**Encoding** UTF-8

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.3.3

**Config/rextendr/version** 0.4.2

**SystemRequirements** Cargo (Rust's package manager), rustc

**Depends** R (>= 4.2)

**Imports** cli, generics, tibble

**Suggests** knitr, rmarkdown, testthat (>= 3.0.0), withr

**Config/testthat/edition** 3

**URL** <http://christophertkenny.com/ipf/>

**VignetteBuilder** knitr

**LazyData** true

**Config/Needs/website** christopherkenny/ctktemplate

**Config/pak/sysreqs** libclang-dev

**Repository** <https://christopherkenny.r-universe.dev>

**Date/Publication** 2026-05-09 20:50:47 UTC

**RemoteUrl** <https://github.com/christopherkenny/ipf>

**RemoteRef** HEAD

**RemoteSha** 739b8b6413ac3cfb6e494bbba61f3014fc8aeb5c

## Contents

anes24 . . . . .	2
augment.ipf_rake . . . . .	3
design_effect . . . . .	3
find_discrepant_vars . . . . .	4
glance.ipf_rake . . . . .	5
print.ipf_rake . . . . .	6
rake . . . . .	6
summary.ipf_rake . . . . .	8
tidy.ipf_rake . . . . .	9
weight_assess . . . . .	10

<b>Index</b>	<b>11</b>
--------------	-----------

---

anes24	<i>ANES 2024 Time Series Study (subset)</i>
--------	---

---

### Description

A subset of the 2024 American National Election Study (ANES) Time Series face-to-face sample, containing demographic and vote choice variables for 966 respondents. Useful for demonstrating survey raking workflows.

### Usage

anes24

### Format

A tibble with 966 rows and 7 columns:

**state** Two-letter US state abbreviation. NA for respondents whose state is not identified (106 missing).

**sex** Respondent sex: "Male" or "Female" (5 missing).

**race** Race/ethnicity: "White", "Black", "Hispanic", "Asian", or "Other" (11 missing).

**income** Household income bracket: "Under \$50k", "\$50k-\$100k", or "Over \$100k" (47 missing).

**education** Education: "Less than HS", "High school", "Some college", "Bachelor's", or "Graduate" (451 missing).

**married** Marital status: "Married", "Widowed", "Divorced", "Separated", or "Never married" (277 missing).

**presidential** 2024 presidential vote choice: "Harris" or "Trump" (335 missing).

### Source

<https://electionstudies.org/data-center/2024-time-series-study/>

## References

American National Election Studies. 2025. *ANES 2024 Time Series Study Full Release* (dataset and documentation). August 8, 2025 version. <https://www.electionstudies.org/>

---

augment.ipf_rake	<i>Augment data with raked weights</i>
------------------	--

---

## Description

Returns the original data frame with a `.weight` column appended.

## Usage

```
## S3 method for class 'ipf_rake'  
augment(x, ...)
```

## Arguments

<code>x</code>	An <code>ipf_rake</code> object.
<code>...</code>	Additional arguments (ignored).

## Value

A tibble with all original columns plus `.weight`.

## Examples

```
data <- data.frame(  
  gender = sample(c('M', 'F'), 100, replace = TRUE, prob = c(0.6, 0.4))  
)  
targets <- list(gender = c(M = 0.5, F = 0.5))  
result <- rake(data, targets)  
augment(result)
```

---

design_effect	<i>Compute design effect and effective sample size</i>
---------------	--

---

## Description

The design effect (`deff`) measures the variance inflation factor due to unequal weighting. The effective sample size is  $n / \text{deff}$ .

## Usage

```
design_effect(weights)
```

**Arguments**

weights          Numeric weight vector.

**Value**

A list with deff (design effect) and n\_eff (effective sample size).

**Examples**

```
w <- c(1.2, 0.8, 1.5, 0.5, 1.0)
design_effect(w)
```

---

find\_discrepant\_vars    *Find discrepant variables and their aggregate discrepancy scores*

---

**Description**

Calculates discrepancy between the current weighted distribution and target distributions for each variable, then aggregates using the chosen method.

**Usage**

```
find_discrepant_vars(
  data,
  targets,
  weights,
  choosemethod = "total",
  na_method = c("exclude", "bucket")
)
```

**Arguments**

data                Data frame.

targets            Named list of named numeric target vectors (proportions).

weights            Numeric weight vector.

choosemethod      Method for aggregating per-category discrepancies. One of "total", "max", "average", "totalsquared", "maxsquared", "averagesquared".

na\_method         How to handle NA values. "exclude" skips NA cases from that margin. "bucket" treats missing values as an implicit extra category.

**Value**

Named numeric vector of aggregate discrepancy per variable.

## Examples

```
data <- data.frame(
  gender = sample(c('M', 'F'), 100, replace = TRUE, prob = c(0.6, 0.4)),
  age = sample(c('young', 'old'), 100, replace = TRUE, prob = c(0.7, 0.3))
)
targets <- list(
  gender = c(M = 0.5, F = 0.5),
  age = c(young = 0.6, old = 0.4)
)
find_discrepant_vars(data, targets, weights = rep(1, 100))
```

---

glance.ipf_rake	<i>Glance at an ipf_rake object</i>
-----------------	-------------------------------------

---

## Description

Returns a single-row tibble with summary statistics.

## Usage

```
## S3 method for class 'ipf_rake'
glance(x, ...)
```

## Arguments

x                    An ipf\_rake object.  
...                   Additional arguments (ignored).

## Value

A single-row tibble with columns: converged, iterations, max\_prop\_err, deff, n\_eff, n\_obs, n\_vars.

## Examples

```
data <- data.frame(
  gender = sample(c('M', 'F'), 100, replace = TRUE, prob = c(0.6, 0.4))
)
targets <- list(gender = c(M = 0.5, F = 0.5))
result <- rake(data, targets)
glance(result)
```

---

<code>print.ipf_rake</code>	<i>Print an ipf_rake object</i>
-----------------------------	---------------------------------

---

**Description**

Print an ipf\_rake object

**Usage**

```
## S3 method for class 'ipf_rake'  
print(x, ...)
```

**Arguments**

<code>x</code>	An ipf_rake object.
<code>...</code>	Additional arguments (ignored).

**Value**

Invisibly returns x.

**Examples**

```
data <- data.frame(  
  gender = sample(c('M', 'F'), 100, replace = TRUE, prob = c(0.6, 0.4))  
)  
targets <- list(gender = c(M = 0.5, F = 0.5))  
result <- rake(data, targets)  
print(result)
```

---

<code>rake</code>	<i>Iterative proportional fitting (raking)</i>
-------------------	--

---

**Description**

Adjusts survey weights so that weighted marginal distributions match known population targets. Supports automatic variable selection, iterative re-raking, and weight bounding.

**Usage**

```
rake(  
  data,  
  targets,  
  base_weights = NULL,  
  cap = 5,  
  bounds = NULL,
```

```

type = c("nolim", "pctlim", "nlim"),
pctlim = 0.05,
nlim = 5L,
choosemethod = c("total", "max", "average", "totalsquared", "maxsquared",
  "averagesquared"),
na_method = c("exclude", "bucket"),
iterate = TRUE,
max_iter = 1000L,
tol = 1e-06,
verbose = FALSE,
diagnostics_every = 0L
)

```

## Arguments

<code>data</code>	A data frame or tibble containing the survey data.
<code>targets</code>	A named list of named numeric vectors specifying target proportions for each raking variable. Names of the list must match column names in <code>data</code> . Each vector's names must match the levels of the corresponding variable. Values should sum to 1 (proportions); if not, they are normalized with a warning.
<code>base_weights</code>	Optional numeric vector of base (design) weights. If <code>NULL</code> (default), uniform weights of 1 are used. Centered to mean 1 before raking.
<code>cap</code>	Maximum weight value (ratio cap). Weights exceeding this value are trimmed and all weights are renormalized. Default 5. Ignored if <code>bounds</code> is specified.
<code>bounds</code>	Optional numeric vector of length 2, <code>c(lo, hi)</code> , specifying minimum and maximum weight bounds. Overrides <code>cap</code> .
<code>type</code>	Variable selection method: <ul style="list-style-type: none"> <li>• <code>"nolim"</code> (default): use all variables in targets.</li> <li>• <code>"pctlim"</code>: use only variables with discrepancy <math>\geq</math> <code>pctlim</code>.</li> <li>• <code>"nlim"</code>: use the <code>nlim</code> most discrepant variables.</li> </ul>
<code>pctlim</code>	Discrepancy threshold for <code>type = "pctlim"</code> . Default 0.05 (5 percentage points).
<code>nlim</code>	Number of variables for <code>type = "nlim"</code> . Default 5.
<code>choosemethod</code>	Method for aggregating per-category discrepancies into a single variable score. One of <code>"total"</code> , <code>"max"</code> , <code>"average"</code> , <code>"totalsquared"</code> , <code>"maxsquared"</code> , <code>"averagesquared"</code> .
<code>na_method</code>	How to handle NA values in raking variables. <code>"exclude"</code> (default): targets are proportions among non-NA cases only; NA cases are invisible to that margin. Matches <code>anesrake</code> . <code>"bucket"</code> : NAs become a frozen extra category; their total weight is preserved and the named targets are rescaled to the remaining non-NA mass.
<code>iterate</code>	Logical. If <code>TRUE</code> and <code>type = "pctlim"</code> , re-check discrepancies after raking and add newly discrepant variables, repeating up to 10 times. Default <code>TRUE</code> .
<code>max_iter</code>	Maximum number of raking iterations. Default 1000.
<code>tol</code>	Convergence tolerance (max proportional error). Default 1e-6.
<code>verbose</code>	Logical. If <code>TRUE</code> , print iteration progress. Default <code>FALSE</code> .

diagnostics\_every

Record per-margin diagnostics every k iterations. 0 means only baseline. Default 0.

### Value

An ipf\_rake object (S3 class) containing:

- weights: final raked weight vector
- data: the input data frame
- converged: logical
- iterations: number of iterations
- max\_prop\_err: final max proportional error
- targets: normalized targets used
- vars\_used: character vector of variables raked on
- base\_weights: original base weights
- type, choosmethod, na\_method, cap: settings used
- deff, n\_eff: design effect and effective sample size
- diagnostics: tibble of per-iteration diagnostics

### Examples

```
data <- data.frame(
  gender = sample(c('M', 'F'), 100, replace = TRUE, prob = c(0.6, 0.4)),
  age = sample(c('young', 'old'), 100, replace = TRUE, prob = c(0.7, 0.3))
)
targets <- list(
  gender = c(M = 0.5, F = 0.5),
  age = c(young = 0.6, old = 0.4)
)
result <- rake(data, targets)
print(result)
```

---

summary.ipf\_rake      *Summarize an ipf\_rake object*

---

### Description

Produces a detailed summary including per-variable diagnostic tables showing target vs. achieved distributions.

### Usage

```
## S3 method for class 'ipf_rake'
summary(object, ...)
```

**Arguments**

object            An ipf\_rake object.  
 ...                Additional arguments (ignored).

**Value**

Invisibly returns a list with convergence info, weight summary, design effect, and per-variable assessment tibbles.

**Examples**

```
data <- data.frame(
  gender = sample(c('M', 'F'), 100, replace = TRUE, prob = c(0.6, 0.4))
)
targets <- list(gender = c(M = 0.5, F = 0.5))
result <- rake(data, targets)
summary(result)
```

---

tidy.ipf_rake	<i>Tidy an ipf_rake object</i>
---------------	--------------------------------

---

**Description**

Returns a one-row-per-variable-per-level tibble with target proportions, weighted proportions, and discrepancy.

**Usage**

```
## S3 method for class 'ipf_rake'
tidy(x, ...)
```

**Arguments**

x                    An ipf\_rake object.  
 ...                Additional arguments (ignored).

**Value**

A tibble with columns: variable, level, target, weighted\_pct, discrepancy.

**Examples**

```
data <- data.frame(
  gender = sample(c('M', 'F'), 100, replace = TRUE, prob = c(0.6, 0.4))
)
targets <- list(gender = c(M = 0.5, F = 0.5))
result <- rake(data, targets)
tidy(result)
```

---

`weight_assess`*Assess weight quality with diagnostic tables*

---

**Description**

Produces a per-variable diagnostic table comparing target distributions to unweighted and weighted distributions.

**Usage**

```
weight_assess(  
  data,  
  targets,  
  weights,  
  base_weights = NULL,  
  na_method = c("exclude", "bucket")  
)
```

**Arguments**

<code>data</code>	Data frame.
<code>targets</code>	Named list of named numeric target vectors (proportions).
<code>weights</code>	Final raked weight vector.
<code>base_weights</code>	Original base weights before raking. If NULL, uses uniform weights.
<code>na_method</code>	How to handle NA values. "exclude" skips NA cases from that margin. "bucket" treats missing values as an implicit extra category.

**Value**

Named list of tibbles, one per variable.

**Examples**

```
data <- data.frame(  
  gender = sample(c('M', 'F'), 100, replace = TRUE, prob = c(0.6, 0.4))  
)  
targets <- list(gender = c(M = 0.5, F = 0.5))  
result <- rake(data, targets)  
weight_assess(data, targets, result$weights)
```

# Index

## \* datasets

anes24, [2](#)

anes24, [2](#)

augment.ipf\_rake, [3](#)

design\_effect, [3](#)

find\_discrepant\_vars, [4](#)

glance.ipf\_rake, [5](#)

print.ipf\_rake, [6](#)

rake, [6](#)

summary.ipf\_rake, [8](#)

tidy.ipf\_rake, [9](#)

weight\_assess, [10](#)