Package: moder (via r-universe)

August 27, 2024

Title Mode Estimation

Version 0.2.1.9000

Description Determines single or multiple modes (most frequent values). Checks if missing values make this impossible, and returns 'NA' in this case. Dependency-free source code. See Franzese and Iuliano (2019) <doi:10.1016/B978-0-12-809633-8.20354-3>.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

Suggests devtools, ggplot2, knitr, rmarkdown, stats, testthat (>= 3.0.0), tibble, utils

Config/testthat/edition 3

URL https://github.com/lhdjung/moder, https://lhdjung.github.io/moder/

BugReports https://github.com/lhdjung/moder/issues

VignetteBuilder knitr

Collate 'counts.R' 'frequencies.R' 'frequency-grid-df.R' 'frequency-grid-plot.R' 'mode-proper.R' 'mode-df.R' 'mode-possible.R' 'predicates.R' 'utils.R'

Repository https://lhdjung.r-universe.dev

RemoteUrl https://github.com/lhdjung/moder

RemoteRef HEAD

RemoteSha 2663e14d11a1a708dc9bce1b2a4fee5ce5bca2d9

Contents

`frequency_grid_df` *Frequency grid data frame*

Description

NOTE: This function is currently experimental and shouldn't be relied upon.

`frequency_grid_df()` takes a vector and creates an extended frequency table about it. Internally, this is used as a basis for `frequency_grid_plot()`.

Usage

```
frequency_grid_df(x, max_unique = NULL)
```

Arguments

`x`	A vector.
`max_unique`	Numeric or string. If the maximum number of unique values in `x` is known, set `max_unique` to that number. This rules out that `NA`s represent values beyond that number (see examples). Set it to `"known"` instead if no values beyond those already known can occur. Default is `NULL`, which assumes no maximum.

Value

A data frame with these columns:

- `x`: The input vector, with each unique known value repeated to be as frequent as the most frequent one.
- `freq` (integer): Hypothetical frequency of each `x` value.
- `is_missing` (logical): Is the observation absent from the input vector?
- `can_be_filled` (logical): Are there enough `NA`s so that one of them might hypothetically represent the `x` value in question, implying that there would be at least as many observations of that value as the respective frequency (`freq`) indicates?
- `is_supermodal` (logical): Is the frequency of this value greater than the maximum frequency among known values?

Limitations

See the limitations section of `frequency_grid_plot()`.

Examples

```
x <- c("a", "a", "a", "b", "b", "c", NA, NA, NA, NA, NA)
frequency_grid_df(x)
```

`frequency_grid_plot` *Frequency grid ggplot*

Description

NOTE: This function is currently experimental and shouldn't be relied upon.

Call `frequency_grid_plot()` to visualize the absolute frequencies of values in a vector. Each observation is plotted distinctly, resulting in a hybrid of a histogram and a scatterplot.

- Boxes are known values.
- Circles with `NA` labels are missing values.
- Empty circles are no values at all: They signify that certain unique values would have to be more frequent in order for all unique values to be equally frequent.

Usage

```
frequency_grid_plot(
  x,
  show_line_grid = FALSE,
  show_line_mode = FALSE,
  label_missing = "NA",
  color_label_missing = "red2",
  color_missing = "red2",
  color_non_missing = "blue2",
  alpha_missing = 1,
  alpha_non_missing = 0.75,
  size_label_missing = 3,
  size_missing = 10,
  size_non_missing = 10,
  shape_missing = 1,
  shape_non_missing = 15,
  expand = 0.1
)
```

Arguments

x	A vector with frequencies to visualize.
show_line_grid	Logical. Should gridlines be present, crossing at each observation? Default is FALSE.
show_line_mode	Logical. Should a dashed line demarcate the mode(s) among known values from the missing values that might add to these modes, if there are any? Default is FALSE.
label_missing	String. Label used for missing values. Default is "NA".
color_label_missing, color_missing, color_non_missing	String. Colors of the data points. Defaults are "red2" for missing data points as well as their labels, and "blue2" for non-missing data points.
alpha_missing, alpha_non_missing	Numeric. Opacity of the data points. Defaults are 1 and 0.75, respectively.
size_label_missing, size_missing, size_non_missing	Numeric. Sizes of the data points. Defaults are 3 for the label and 10 for both symbols.
shape_missing, shape_non_missing	Numeric or string. Signifiers for the shapes of the data points. Defaults are 1 (circle) and 15 (square filled), respectively.
expand	Numeric. Padding whitespace between the axes and the data points. The distance is the same on all four sides due to the grid structure. Default is 0.1.

Value

A ggplot object. To save it, call ggplot2::ggsave().

Limitations

Certain assumptions about missing values are currently hard-coded in the function. In the future, they should become optional. These assumptions are:

- All missings represent a known value. For example, in c(1, 2, NA), the NA is either 1 or 2.
- The missings are as evenly distributed across known values as possible. Therefore, in c(1, 2, NA, NA), one NA is a 1 and the other one is a 2. This is clearly not reasonable as a general assumption. It is derived from moder's way of determining possible extreme cases.

See Also

frequency_grid_df(), which forms the basis of the current function.

Examples

```
x <- c("a", "a", "a", "b", "b", "c", NA, NA, NA, NA, NA)

# Basic usage:
frequency_grid_plot(x)
```

```
# With "N/A" as a marker of missing values
# instead of "NA":
frequency_grid_plot(x, label_missing = "N/A")

# Black and white mode:
frequency_grid_plot(
  x, color_label_missing = "black",
  color_missing = "black", color_non_missing = "black"
)
```

`mode-possible` *Possible sets of modes*

Description

`mode_possible_min()` and `mode_possible_max()` determine the minimal and maximal sets of modes, given the number of missing values.

Usage

```
mode_possible_min(x, accept = FALSE, multiple = NULL)

mode_possible_max(x, accept = FALSE, multiple = NULL)
```

Arguments

`x`	A vector to search for its possible modes.
`accept`	Logical. If `accept` is set to `TRUE`, the functions don't necessarily return the minimal or maximal sets of modes but all values that *might* be part of those sets. Default is `FALSE`. See details.
`multiple`	Deprecated; will be removed in the future. Old name of `accept`.

Details

If `accept = TRUE`, the functions return multiple values that may or may not be modes depending on the true values behind `NA`s. Why is this disabled by default? In cases where multiple unique values would be modes if and only if one or more missing values represented them but there are not enough missing values to represent all of them, any values that are not represented by enough `NA`s would not be modes. This makes it unclear which unique values are part of the minimal and maximal sets of modes, so the default of `accept` is to return `NA` in these cases.

Value

Vector of the same type as `x`. By default, it contains the minimal or maximal possible sets of modes (values tied for most frequent) in `x`. If the functions can't determine these possible modes because of missing values, they return `NA`.

See Also

`mode_count_range()` for the minimal and maximal *numbers* of possible modes. They can always be determined, even if the present functions return `NA`.

Examples

```
# "a" is guaranteed to be a mode,
# "b" might also be one, but
# "c" is impossible:
mode_possible_min(c("a", "a", "a", "b", "b", "c", NA))
mode_possible_max(c("a", "a", "a", "b", "b", "c", NA))

# Only `8` can possibly be the mode
# because, even if `NA` is `7`, it's
# still less frequent than `8`:
mode_possible_min(c(7, 7, 8, 8, 8, 8, NA))
mode_possible_max(c(7, 7, 8, 8, 8, 8, NA))

# No clear minimal or maximal set
# of modes because `NA` may tip
# the balance between `1` and `2`
# towards a single mode:
mode_possible_min(c(1, 1, 2, 2, 3, 4, 5, NA))
mode_possible_max(c(1, 1, 2, 2, 3, 4, 5, NA))

# With `accept = TRUE`, the functions
# return all values that might be part of
# the min / max sets of modes; not these
# sets themselves:
mode_possible_min(c(1, 1, 2, 2, 3, 4, 5, NA), accept = TRUE)
mode_possible_max(c(1, 1, 2, 2, 3, 4, 5, NA), accept = TRUE)
```

`mode_all` *All modes*

Description

`mode_all()` returns the set of all modes in a vector.

Usage

```
mode_all(x, na.rm = FALSE, na.rm.amount = 0)
```

Arguments

`x`	A vector to search for its modes.
`na.rm`	Logical. Should missing values in `x` be removed before computation proceeds? Default is `FALSE`.
`na.rm.amount`	Numeric. Alternative to `na.rm` that only removes a specified number of missing values. Default is `0`.

Value

A vector with all modes (values tied for most frequent) in `x`. If the modes can't be determined because of missing values, returns `NA` instead.

See Also

- `mode_first()` for the first-appearing mode.
- `mode_single()` for the *only* mode, or `NA` if there are more.

Examples

```
# Both `3` and `4` are the modes:
mode_all(c(1, 2, 3, 3, 4, 4))

# Only `8` is:
mode_all(c(8, 8, 9))

# Can't determine the modes here --
# `9` might be another mode:
mode_all(c(8, 8, 9, NA))

# Either `1` or `2` could be a
# single mode, depending on `NA`:
mode_all(c(1, 1, 2, 2, NA))

# `1` is the most frequent value,
# no matter what `NA` stands for:
mode_all(c(1, 1, 1, 2, NA))

# Ignore `NA`s with `na.rm = TRUE`
# (there should be good reasons for this!):
mode_all(c(8, 8, 9, NA), na.rm = TRUE)
mode_all(c(1, 1, 2, 2, NA), na.rm = TRUE)
```

---

mode_count *Modal count*

Description

`mode_count()` counts the modes in a vector. Thin wrapper around `mode_all()`.

Usage

```
mode_count(x, na.rm = FALSE, max_unique = NULL)
```

Arguments

x	A vector to search for its modes.
na.rm	Logical. Should missing values in x be removed before computation proceeds? Default is FALSE.
max_unique	Numeric or string. If the maximum number of unique values in x is known, set max_unique to that number. This rules out that NAs represent values beyond that number (see examples). Set it to "known" instead if no values beyond those already known can occur. Default is NULL, which assumes no maximum.

Value

Integer. Number of modes (values tied for most frequent) in x. If the modes can't be determined because of missing values, returns NA instead.

Examples

```
# There are two modes, `3` and `4`:
mode_count(c(1, 2, 3, 3, 4, 4))

# Only one mode, `8`:
mode_count(c(8, 8, 9))

# Can't determine the number of modes
# here -- `9` might be another mode:
mode_count(c(8, 8, 9, NA))

# Either `1` or `2` could be a
# single mode, depending on `NA`:
mode_count(c(1, 1, 2, 2, NA))

# `1` is the most frequent value,
# no matter what `NA` stands for:
mode_count(c(1, 1, 1, 2, NA))

# Ignore `NA`s with `na.rm = TRUE`
# (there should be good reasons for this!):
mode_count(c(8, 8, 9, NA), na.rm = TRUE)
mode_count(c(1, 1, 2, 2, NA), na.rm = TRUE)
```

---

mode_count_range *Modal count range*

Description

mode_count_range() determines the minimal and maximal number of modes given the number of missing values.

Usage

```
mode_count_range(x, max_unique = NULL)
```

Arguments

x	A vector to search for its possible modes.
max_unique	Numeric or string. If the maximum number of unique values in x is known, set max_unique to that number. This rules out that NAs represent values beyond that number (see examples). Set it to "known" instead if no values beyond those already known can occur. Default is NULL, which assumes no maximum.

Details

If x is a factor, max_unique should be "known" or there is a warning. This is because a factor's levels are supposed to include all of its possible values.

Value

Integer (length 2). Minimal and maximal number of modes (values tied for most frequent) in x.

Examples

```
# If `NA` is `7` or `8`, that number is
# the only mode; otherwise, both numbers
# are modes:
mode_count_range(c(7, 7, 8, 8, NA))

# Same result here -- `7` is the only mode
# unless `NA` is secretly `8`, in which case
# there are two modes:
mode_count_range(c(7, 7, 7, 8, 8, NA))

# But now, there is no way for `8` to be
# as frequent as `7`:
mode_count_range(c(7, 7, 7, 7, 8, 8, NA))

# The `NA`s might form a new mode here
# if they are both, e.g., `9`:
mode_count_range(c(7, 7, 8, 8, NA, NA))

# However, if there can be no values beyond
# those already known -- `7` and `8` --
# the `NA`s can't form a new mode.
# Specify this with `max_unique = "known"`:
mode_count_range(c(7, 7, 8, 8, NA, NA), max_unique = "known")
```

mode_df *Tabulate mode estimates with the certainty about them*

Description

`mode_df()` takes a data frame (or another list) of numeric vectors and computes the mode or modes of each element. Where the true mode is unknown due to missing values, more and more `NA`s are ignored until an estimate for the mode is found.

Estimates are presented along with information about whether they are known to be the true mode, how many `NA`s had to be ignored during estimation, the rate of ignored `NA`s, etc.

Usage

```
mode_df(
  x,
  method = c("first", "all", "single"),
  na.rm.from = c("first", "last", "random"),
  accept = FALSE,
  multiple = c("NA", "min", "max", "mean", "median", "first", "last", "random")
)
```

Arguments

`x`	List of vectors. Each vector needs to be numeric or similar. Note that data frames are lists, so `x` can be a data frame.
`method`	String. How to determine the mode(s)? Options are: • `"first"` for `mode_first()`, the default. • `"all"` for `mode_all()`. This may return multiple values per estimate. • `"single"` for `mode_single()`. The only option that can return `NA` estimates.
`na.rm.from`	String. Only relevant to the default `method = "first"`. Where to start when removing `NA`s from `x`? Options are `"start"`, `"end"`, and `"random"`. Default is `"start"`.
`accept`	Passed on to `mode_first()` and `mode_single()`. Default is `FALSE`.
`multiple`	Passed on to `mode_single()`. Default is `"NA"`.

Details

The function deals with missing values (`NA`s) by first checking whether they make the true mode unknown. If they do, it removes one `NA`, then checks again; and so on until an estimate is found.

This strategy is based on the `na.rm.amount` argument of `mode_first()`, `mode_all()`, and `mode_single()`. It represents a middle way between simply ignoring all `NA`s and not even trying to compute an estimate. Instead, it only removes the minimum number of `NA`s necessary, because some distributions have a known mode (or set of modes) even if some of their values are missing. By keeping track of the removed `NA`s, `mode_df()` quantifies the uncertainty about its estimates.

Value

Tibble (data frame) with these columns:

- `term`: the names of `x` elements. Only present if any are named.
- `estimate`: the modes of `x` elements, ignoring as many `NA`s as necessary. List-column if `method = "all"`.
- `certainty`: `TRUE` if the corresponding estimate is certain to be the true mode, and `FALSE` if this is unclear due to missing values.
- `na_ignored`: the number of missing values that had to be ignored to arrive at the estimate.
- `na_total`: the total number of missing values.
- `rate_ignored_na`: the proportion of missing values that had to be ignored from among all missing values.
- `sum_total`: the total number of values, missing or not.
- `rate_ignored_sum`: the proportion of missing values that had to be ignored from among all values, missing or not.

Examples

```
# Use a list of numeric vectors:
my_list <- list(
  a = 1:15,
  b = c(1, 1, NA),
  c = c(4, 4, NA, NA, NA, NA),
  d = c(96, 24, 3, NA)
)

mode_df(my_list)

# Data frames are allowed:
mode_df(iris[1:4])
```

---

mode_first *The first-appearing mode*

Description

`mode_first()` returns the mode that appears first in a vector, i.e., before any other modes.

Usage

```
mode_first(
  x,
  na.rm = FALSE,
  na.rm.amount = 0,
  na.rm.from = c("first", "last", "random"),
  accept = FALSE
)
```

Arguments

x	A vector to search for its first mode.
na.rm	Logical. Should missing values in x be removed before computation proceeds? Default is FALSE.
na.rm.amount	Numeric. Alternative to na.rm that only removes a specified number of missing values. Default is 0.
na.rm.from	String. If na.rm.amount is used, from which position in x should missing values be removed? Options are "first", "last", and "random". Default is "first".
accept	Logical. Should the first-appearing value known to be a mode be accepted? If FALSE (the default), returns NA if a value that appears earlier might be another mode due to missing values.

Details

Unlike mode_all() and mode_single(), mode_first() has an na.rm.from argument. That is because it cares about the order of x values, whereas the other ones do not.

Value

The first mode (most frequent value) in x. If it can't be determined because of missing values, returns NA instead.

See Also

- mode_all() for the full set of modes.
- mode_single() for the *only* mode, or NA if there are more.

Examples

```
# `2` is most frequent:
mode_first(c(1, 2, 2, 2, 3))

# Can't determine the first mode --
# it might be `1` or `2` depending
# on the true value behind `NA:
mode_first(c(1, 1, 2, 2, NA))

# Ignore `NA`s with `na.rm = TRUE`
# (there should be good reasons for this!):
mode_first(c(1, 1, 2, 2, NA), na.rm = TRUE)

# `1` is the most frequent value,
# no matter what `NA` stands for:
mode_first(c(1, 1, 1, 2, NA))

# By default, the function insists on
# the first mode, so it won't accept the
# first value *known* to be a mode if an
# earlier value might be a mode, too:
```

```
mode_first(c(1, 2, 2, NA))

# You may accept the first-known mode:
mode_first(c(1, 2, 2, NA), accept = TRUE)
```

`mode_frequency` *Modal frequency*

Description

Call `mode_frequency()` to get the number of times that a vector's mode appears in the vector.

See `mode_frequency_range()` for bounds on an unknown frequency.

Usage

```
mode_frequency(x, na.rm = FALSE, max_unique = NULL)
```

Arguments

`x`	A vector to check for its modal frequency.
`na.rm`	Logical. Should missing values in `x` be removed before computation proceeds? Default is `FALSE`.
`max_unique`	Numeric or string. If the maximum number of unique values in `x` is known, set `max_unique` to that number. This rules out that `NA`s represent values beyond that number (see examples). Set it to `"known"` instead if no values beyond those already known can occur. Default is `NULL`, which assumes no maximum.

Details

By default (`na.rm = FALSE`), the function returns `NA` if any values are missing. That is because missings make the frequency uncertain even if the mode is known: any missing value may or may not be the mode, and hence count towards the modal frequency.

Value

Integer (length 1) or `NA`.

See Also

`mode_first()`, which the function wraps.

Examples

```
# The mode, `9`, appears three times:
mode_frequency(c(7, 8, 8, 9, 9, 9))

# With missing values, the frequency
# is unknown, even if the mode isn't:
mode_frequency(c(1, 1, NA))

# You can ignore this problem and
# determine the frequency among known values
# (there should be good reasons for this!):
mode_frequency(c(1, 1, NA), na.rm = TRUE)
```

---

mode_frequency_range *Modal frequency range*

Description

`mode_frequency_range()` determines the minimum and maximum number of times that a vector's mode appears in the vector. The minimum assumes that no `NA`s are the mode; the maximum assumes that all `NA`s are.

Usage

```
mode_frequency_range(x, max_unique = NULL)
```

Arguments

`x`	A vector to check for its modal frequency.
`max_unique`	Numeric or string. If the maximum number of unique values in `x` is known, set `max_unique` to that number. This rules out that `NA`s represent values beyond that number (see examples). Set it to `"known"` instead if no values beyond those already known can occur. Default is `NULL`, which assumes no maximum.

Details

If there are no `NA`s in `x`, the two return values are identical. If all `x` values are `NA`, the return values are `1` (no two `x` values are the same) and the total number of values (all `x` values are the same).

Value

Integer (length 2).

See Also

`mode_frequency()`, for the precise frequency (or `NA` if it can't be determined).

Examples

```
# The mode is `7`. It appears four or
# five times because the `NA` might
# also be a `7`:
mode_frequency_range(c(7, 7, 7, 7, 8, 8, NA))

# All of `"c"`, `"d"`, and `"e"` are the modes,
# and each of them appears twice:
mode_frequency_range(c("a", "b", "c", "c", "d", "d", "e", "e"))
```

---

mode_is_trivial *Is the mode trivial?*

Description

`mode_is_trivial()` checks whether all values in a given vector are equally frequent. The mode is not too informative in such cases.

Usage

```
mode_is_trivial(x, na.rm = FALSE, max_unique = NULL)
```

Arguments

`x`	A vector to search for its modes.
`na.rm`	Logical. Should missing values in `x` be removed before computation proceeds? Default is `FALSE`.
`max_unique`	Numeric or string. If the maximum number of unique values in `x` is known, set `max_unique` to that number. This rules out that `NA`s represent values beyond that number (see examples). Set it to `"known"` instead if no values beyond those already known can occur. Default is `NULL`, which assumes no maximum.

Details

The function returns `TRUE` whenever `x` has length < 3 because no value is more frequent than another one. Otherwise, it returns `NA` in these cases:

- Some `x` values are missing, all known values are equal, and the number of missing values is divisible by the number of unique known values. Thus, the missings don't necessarily break the tie among known values, and it is unknown whether there is a value with a different frequency.
- All known values are modes if the `NA`s "fill up" the non-modal values exactly, i.e., without any `NA`s remaining.
- Some `NA`s remain after "filling up" the non-modal values with `NA`s (so that they are hypothetically modes), and the number of remaining `NA`s is divisible by the number of unique known values.

- There are so many missing values that they might form mode-sized groups of values that are not among the known values, and the number of NAs is divisible by the modal frequency so that all (partly hypothetical) values might be equally frequent. You can limit the number of such hypothetical values by specifying max_unique. The function might then return FALSE instead of NA.

Value

Logical (length 1).

Examples

```
# The mode is trivial if
# all values are equal...
mode_is_trivial(c(1, 1, 1))

# ...and even if all unique
# values are equally frequent:
mode_is_trivial(c(1, 1, 2, 2))

# It's also trivial if
# all values are different:
mode_is_trivial(c(1, 2, 3))

# Here, the mode is nontrivial
# because `1` is more frequent than `2`:
mode_is_trivial(c(1, 1, 2))

# Two of the `NA`s might be `8`s, and
# the other three might represent a value
# different from both `7` and `8`. Thus,
# it's possible that all three distinct
# values are equally frequent:
mode_is_trivial(c(7, 7, 7, 8, rep(NA, 5)))

# The same is not true if all values,
# even the missing ones, must represent
# one of the known values:
mode_is_trivial(c(7, 7, 7, 8, rep(NA, 5)), max_unique = "known")
```

---

mode_single *The single mode*

Description

mode_single() returns the only mode in a vector. If there are multiple modes, it returns NA by default.

Usage

```
mode_single(
  x,
  na.rm = FALSE,
  na.rm.amount = 0,
  accept = FALSE,
  multiple = c("NA", "min", "max", "mean", "median", "first", "last", "random")
)
```

Arguments

x	A vector to search for its mode.
na.rm	Logical. Should missing values in x be removed before computation proceeds? Default is FALSE.
na.rm.amount	Numeric. Alternative to na.rm that only removes a specified number of missing values. Default is 0.
accept	Logical. Should the minimum set of modes be accepted to check for a single mode? If FALSE (the default), insists on the complete set and returns NA if it can't be determined.
multiple	String or integer (length 1), or a function. What to do if x has multiple modes? The default returns NA. All other options rely on the modal values: "min", "max", "mean", "median", "first", "last", and "random". Alternatively, multiple can be an index number, or a function that summarizes the modes. See details.

Details

If accept is FALSE (the default), the set of modes is obtained via mode_all() instead of mode_possible_min(). Set it to TRUE to avoid returning NA when some, though not all modes are known. The purpose of the default is to insist on a single mode.

If x is a string vector and multiple is "min" or "max", the mode is selected lexically, just like min(letters) returns "a". The "mean" and "median" options return NA with a warning. For factors, "min", "max", and "median" are errors, but "mean" returns NA with a warning. These are inconsistencies in base R.

The multiple options "first" and "last" always select the mode that appears first or last in x. Index numbers, like multiple = 2, allow you to select more flexibly. If multiple is a function, its output must be length 1.

Value

The only mode (most frequent value) in x. If it can't be determined because of missing values, NA is returned instead. If there are multiple modes, NA is returned by default (multiple = "NA").

See Also

- mode_first() for the first-appearing mode.
- mode_all() for the complete set of modes.
- mode_possible_min() for the minimal set of modes.

Examples

```
# `8` is the only mode:
mode_single(c(8, 8, 9))

# With more than one mode, the function
# returns `NA`:
mode_single(c(1, 2, 3, 3, 4, 4))

# Can't determine the modes here --
# `9` might be another mode:
mode_single(c(8, 8, 9, NA))

# Accept `8` anyways if it's
# sufficient to just have any mode:
mode_single(c(8, 8, 9, NA), accept = TRUE)

# `1` is the most frequent value,
# no matter what `NA` stands for:
mode_single(c(1, 1, 1, 2, NA))

# Ignore `NA`s with `na.rm = TRUE`
# (there should be good reasons for this!):
mode_single(c(8, 8, 9, NA), na.rm = TRUE)
```

Index