Package 'moder'

July 23, 2025

Title Mode Estimation

Version 0.2.1

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

RoxygenNote 7.2.3

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

NeedsCompilation no

Author Lukas Jung [aut, cre]

Maintainer Lukas Jung <jung-lukas@gmx.net>

Repository CRAN

Date/Publication 2023-05-07 10:30:02 UTC

Contents

`mode-possible` *Possible sets of modes*

Description

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

Usage

```
mode_possible_min(x, multiple = FALSE)

mode_possible_max(x, multiple = FALSE)
```

Arguments

`x`	A vector to search for its possible modes.
`multiple`	Boolean. If `multiple` is set to `TRUE`, the functions return multiple modes with the same frequency, even if some values are missing. Default is `FALSE` because `NA`s may tip the balance between values that are equally frequent among the known values. Thus, if `multiple = TRUE`, the functions don't necessarily return the minimal or maximal sets of modes, but all values that *might* be part of those sets.

Value

By default, a vector with 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` by default (`multiple = FALSE`).

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 `multiple = 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), multiple = TRUE)
mode_possible_max(c(1, 1, 2, 2, 3, 4, 5, NA), multiple = TRUE)
```

`mode_all` *All modes*

Description

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

Usage

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

Arguments

`x`	A vector to search for its modes.
`na.rm`	Boolean. Should missing values in `x` be removed before computation proceeds? Default is `FALSE`.

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`	Boolean. 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.

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 `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 `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 now 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_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, accept = FALSE)
```

Arguments

`x`	A vector to search for its first mode.
`na.rm`	Boolean. Should missing values in `x` be removed before computation proceeds? Default is `FALSE`.
`accept`	Boolean. 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.

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	Boolean. 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.

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 NAs are the mode; the maximum assumes that all NAs 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 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 there are no NAs 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	Boolean. 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.

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 and all known values are equal. Thus, it is unknown whether there is a value with a different frequency.
- All known values are modes if the NAs "fill up" the non-modal values exactly, i.e., without any NAs remaining.
- Some NAs remain after "filling up" the non-modal values with NAs (so that they are hypothetically modes), and the number of remaining NAs 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

Boolean (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, accept = FALSE, multiple = "NA")
```

Arguments

`x`	A vector to search for its mode.
`na.rm`	Boolean. Should missing values in `x` be removed before computation proceeds? Default is `FALSE`.
`accept`	Boolean. 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. By default, `NA` is also returned if there are multiple modes (`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