Package ‘statacompat’

November 18, 2026

Title Reproduce Selected 'Stata' Expression and Missing-Value Semantics

Version 1.0

Date 2026-10-30

Description A lightweight compatibility layer for researchers translating 'Stata' data-management code into R. Provides vectorised helpers that reproduce selected 'Stata' expression semantics where they differ from base R, most importantly ``Stata's'' rule that numeric missing values are larger than every number: relational comparisons, inlist(), inrange(), cond(), missing(), and the 'egen' row functions rowtotal(), rowmean(), rowmiss() and rownonmiss(). Only the ordinary missing value is fully supported; ``Stata's'' extended missing values are detected and rejected wherever they would change a result.

License MIT + file LICENSE

Encoding UTF-8

Depends R (>= 4.1.0)

Suggests dplyr, haven, knitr, rmarkdown, testthat (>= 3.0.0), withr

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Config/roxygen2/version 8.1.0

URL https://github.com/aripurwantosp/statacompat

BugReports https://github.com/aripurwantosp/statacompat/issues

Author Ari Purwanto Sarwo Prasojo [aut, cre] (ORCID: <https://orcid.org/0000-0002-4862-5523>)

Maintainer Ari Purwanto Sarwo Prasojo <ari.prasojo18@gmail.com>

Repository CRAN

Date/Publication 2026-11-18 09:40:02 UTC

Contents

stata_comparison 2
stata_cond 4
stata_inlist 5
stata_inrange 7
stata_missing 8
stata_row 9

Index **12**

`stata_comparison` *Stata-style relational comparisons*

Description

Vectorised equivalents of Stata's relational operators ==, !=, >, >=, < and <=. Unlike R's operators, they never return NA: following Stata, a numeric missing value is treated as larger than every number, and two missing values are equal.

Usage

```
stata_eq(x, y)

stata_ne(x, y)

stata_gt(x, y)

stata_ge(x, y)

stata_lt(x, y)

stata_le(x, y)
```

Arguments

x, y	Vectors to compare. See *Types* and *Lengths*.

Value

A logical vector without NAs, of the common length of x and y. Use `as.integer()` for a Stata-style 0/1 indicator.

Semantics

In Stata, `. > 1` is true and `. == .` is true (Stata manual [U] 13.2.3: "A missing value is greater than any nonmissing value"). In R, `NA > 1` is `NA`, and code such as `x > 1` silently drops or propagates missing rows.

Stata	R base	statacompat	result
`. == 1`	`NA == 1`	`stata_eq(NA, 1)`	FALSE
`. != 1`	`NA != 1`	`stata_ne(NA, 1)`	TRUE
`. > 1`	`NA > 1`	`stata_gt(NA, 1)`	TRUE
`. >= 1`	`NA >= 1`	`stata_ge(NA, 1)`	TRUE
`. < 1`	`NA < 1`	`stata_lt(NA, 1)`	FALSE
`. <= 1`	`NA <= 1`	`stata_le(NA, 1)`	FALSE
`. == .`	`NA == NA`	`stata_eq(NA, NA)`	TRUE
`. != .`	`NA != NA`	`stata_ne(NA, NA)`	FALSE

Every row follows from the documented ordering of missing values ([U] 12.2.1) and from the rule that relational operators never return missing ([U] 13.2.3).

Missing values. `NA` and `NaN` both stand for Stata's system missing value `.`. `Inf` and `-Inf` are ordinary (very large / very small) numbers; Stata has no infinities (`1/0` is `.` in Stata).

Extended missing values. Stata orders `. < .a < .b < ... < .z` and treats them as distinct. Extended missing values imported with `haven::read_dta()` are R `NA`s carrying a tag. Comparing an extended missing value with a *non-missing* value gives the Stata result; comparing it with a *different* missing value is an error, because version 1.0 does not reproduce the ordering among missing values.

Strings. Strings are compared by their UTF-8 bytes, as in Stata (Stata manual [U] 12.4.2), so `"a" > "B"` is `TRUE` regardless of the R session's locale. Stata's missing string is `""`, which is simply the smallest string. `NA_character_` has no Stata equivalent and is an error.

Types. `x` and `y` must be of the same kind: numbers (double, integer, logical, numeric `haven_labelled`), strings, `Dates`, or `POSIXct` date-times. Mixing kinds is an error, as in Stata. A bare `NA` may be compared with numbers, dates or date-times. Factors are rejected.

Lengths. `x` and `y` must have the same length, or one of them must have length 1. Any other combination is an error (no silent recycling).

See Also

`stata_missing()` to test for missing values, `stata_inrange()` for closed ranges that exclude missing values.

Examples

```
income <- c(5e6, 2e7, NA)

# Stata: gen high_income = income > 10000000
income > 1e7              # base R: NA for the missing income
stata_gt(income, 1e7)     # Stata: the missing income counts as "high"
```

```
# The usual Stata idiom `income > 10000000 & income < .`:
stata_gt(income, 1e7) & stata_lt(income, NA)

stata_eq(c(1, NA), NA)     # Stata: x == .
stata_gt("a", "B")         # byte order, as in Stata
```

`stata_cond` *Stata's* `cond()`*: vectorised if-else with Stata truth rules*

Description

`stata_cond(condition, yes, no, missing)` reproduces Stata's `cond(x, a, b[, c])`: it returns `yes` where `condition` is true, `no` where it is false and, if supplied, `missing` where `condition` is a missing number.

Usage

```
stata_cond(condition, yes, no, missing)
```

Arguments

`condition`	A numeric or logical vector (numeric `haven_labelled` allowed). Strings, factors and dates are rejected.
`yes`, `no`	Values used where `condition` is true or false.
`missing`	Optional value used where `condition` is a missing number. If omitted, such rows get `yes`, as in Stata.

Value

A vector of the common length of the inputs. Each input must have length 1 or that common length.

How the condition is read

Stata reads a numeric condition as **false if zero and true otherwise, and a missing value is "otherwise"**. `stata_cond()` follows that rule for numeric conditions (Stata manual [FN] Programming functions, `cond()`):

`condition`	3 arguments	with `missing = c`
nonzero number, `Inf`	`yes`	`yes`
`0`	`no`	`no`
`NA` / `NaN` (Stata `.`)	`yes`	`c`
`TRUE`	`yes`	`yes`
`FALSE`	`no`	`no`
logical `NA`	error	error

This is **not** what `ifelse()` or `dplyr::if_else()` do: both return `NA` for a missing condition.

A **logical** NA **condition is an error**. In R it almost always comes from an ordinary R comparison such as x < 2 with missing x, and there is no safe way to guess what Stata would have done: Stata's cond(x < 2, a, b) gives b for a missing x (because . < 2 is false), whereas reading the NA produced by R's x < 2 as Stata's missing value would give a. Build the condition with the stata_*() comparisons, which never return NA, or convert it with as.numeric() if you really want a missing condition to count as true.

Stata ignores the fourth argument when the condition is a relational expression, because such expressions are never missing. The same holds here when the condition comes from a stata_*() predicate.

Result type

yes, no and missing must all be numbers (logical, integer, double, numeric haven_labelled), all strings, or all dates of one class. A bare NA branch is allowed with numbers and dates. The result is logical if every branch is logical, integer if every branch is integer or logical, and double otherwise; strings give character, dates keep their class (date-times take the time zone of the first date-time branch). Value labels are dropped. Mixing numbers and strings is an error, as in Stata.

Examples

```
r10a <- c(1, 2, NA)
r10b <- c(2, 1, NA)

# Stata: gen atwork = cond(r10a == 1 | r10b == 1, 1, 0)
stata_cond(stata_eq(r10a, 1) | stata_eq(r10b, 1), 1, 0)

# Numeric conditions: missing counts as true unless `missing` is given
wage <- c(0, 1500, NA)
stata_cond(wage, "paid", "unpaid")
stata_cond(wage, 1, 0, missing = NA)  # Stata: cond(wage, 1, 0, .)

# A logical NA is rejected:
try(stata_cond(wage > 0, 1, 0))
```

---

stata_inlist *Stata's* inlist()*: is a value equal to any of the candidates?*

Description

stata_inlist(x, a, b, ...) reproduces Stata's inlist(z, a, b, ...): it is TRUE where x equals at least one candidate, using the same equality as stata_eq().

Usage

```
stata_inlist(x, ...)
```

Arguments

x	A vector.
...	One or more candidate values, each of length 1 or `length(x)`.

Value

A logical vector without NAs.

Semantics

- **Missing values are ordinary members.** `stata_inlist(NA, 1, 2)` is `FALSE` and `stata_inlist(NA, 1, NA)` is `TRUE`. R's `NA == 1` would instead give `NA`. Stata's manual does not discuss missing values in `inlist()` explicitly; this rule is inferred from the fact that `inlist()` uses the same equality as `==`, under which `. == .` is true ([U] 13.2.3).
- **Candidates are evaluated row by row.** Each candidate may be a single value or a vector as long as `x` (a "variable"). With a variable candidate, row `i` of `x` is compared with row `i` of the candidate, exactly as Stata evaluates `inlist(x, a, b)` observation by observation.
- **A vector of codes is not a set.** `stata_inlist(x, codes)` with `codes <- c(1, 2, 3)` does *not* ask whether `x` is one of 1, 2, 3; it is an error unless `length(codes)` is 1 or `length(x)`, in which case `codes` is treated as a variable. To test membership in a vector of codes, splice it: `do.call(stata_inlist, c(list(x), as.list(codes)))`.
- **Types.** All arguments must be numbers, or all strings (Stata's rule), or all dates of one class. `NA_character_` is an error; Stata's missing string is `""`, which is an ordinary member.
- **Extended missing values.** Comparing an extended missing value (`.a`-`.z`, see stata_comparison) with a different missing value is an error.
- Stata limits `inlist()` to 250 numeric or 10 string arguments. That is a Stata implementation limit and is not enforced here.

See Also

`stata_eq()`, `stata_inrange()`

Examples

```
kbli <- c(1, 4, NA, 3)
stata_inlist(kbli, 1, 2, 3)
stata_inlist(kbli, 1, NA)     # Stata: inlist(kbli, 1, .)

# A vector of codes must be spliced:
codes <- c(1, 2, 3)
do.call(stata_inlist, c(list(kbli), as.list(codes)))

stata_inlist(c("a", "", "c"), "a", "")
```

`stata_inrange` *Stata's* `inrange()`*: is a value known to lie in a closed interval?*

Description

`stata_inrange(x, lower, upper)` reproduces Stata's `inrange(z, a, b)`, which is `TRUE` "if it is known that a <= z <= b" (Stata manual [FN]). Both bounds are inclusive, a missing `x` is never in range, and a **missing bound means "unbounded" on that side**.

Usage

```
stata_inrange(x, lower, upper)
```

Arguments

`x`	A vector.
`lower`, `upper`	Inclusive bounds; `NA` (or `""` for strings) means unbounded.

Value

A logical vector without `NA`s.

Semantics

Stata applies these rules in order (Stata manual [FN] Programming functions):

1. `x` missing: `FALSE`.
2. `lower` and `upper` missing: `TRUE`.
3. `lower` missing: `x <= upper`.
4. `upper` missing: `lower <= x`.
5. Otherwise: `lower <= x & x <= upper`.

So `stata_inrange(age, 15, NA)` means "age 15 or older, and not missing", which differs from both `age >= 15` in Stata (true for missing age) and `age >= 15` in R (`NA` for missing age). Reversed bounds give `FALSE`. Because rules 2-4 only ask *whether* a bound is missing, extended missing values (`.a`-`.z`) behave exactly like `.` here.

Strings. `""` is Stata's missing string: a `""` value of `x` is never in range, and a `""` bound is unbounded. Strings are ordered by their UTF-8 bytes. Stata's manual adds that for strings `"."` is interpreted as `""`. statacompat does **not** implement that rule, so `"."` is treated as an ordinary string. `NA_character_` is an error.

Types and lengths. `x`, `lower` and `upper` must be of one kind (numbers, strings, or dates of one class; a bare `NA` bound is allowed with numbers and dates). Each must have length 1 or a common length; bounds may therefore vary by row.

See Also

`stata_inlist()`, stata_comparison

Examples

```
age <- c(10, 15, 64, 65, NA)
stata_inrange(age, 15, 64)

# Open-ended range: 15 and older, missing excluded
stata_inrange(age, 15, NA)

# Compare: Stata's `age >= 15` is TRUE for the missing age
stata_ge(age, 15)

stata_inrange(c("m", "", "z"), "a", "n")
```

---

stata_missing *Stata's* missing()*: is any argument missing?*

Description

`stata_missing(x1, x2, ...)` reproduces Stata's `missing(x1, x2, ...)`: `TRUE` where at least one argument is missing.

Usage

```
stata_missing(...)
```

Arguments

...	One or more vectors.

Value

A logical vector without `NA`s.

What counts as missing

- **Numbers and dates:** `NA` and `NaN`, including extended missing values from `haven::tagged_na()` (Stata `.`, `.a`-`.z`). `Inf` is not missing.
- **Strings:** the empty string `""`, Stata's missing string. A string of spaces such as `" "` is **not** missing, because `""` is Stata's only missing string. `NA_character_`, which cannot occur in Stata data, is also counted as missing so that data created in R are handled sensibly.

Arguments may mix numbers and strings, as in Stata. Each must have length 1 or a common length. Factors are rejected.

See Also

`stata_rowmiss()` to count missing values per row.

Examples

```
stata_missing(c(1, NA, NaN, Inf))
stata_missing(c("a", "", " "))

# TRUE where either is missing (Stata: missing(age, name))
stata_missing(c(30, NA, 40), c("Ani", "Budi", ""))
```

---

stata_row *Stata's* egen *row functions*

Description

Row-wise totals, means and missing-value counts across several variables, reproducing egen `rowtotal()`, `rowmean()`, `rowmiss()` and `rownonmiss()`.

Usage

```
stata_rowtotal(..., missing = FALSE)

stata_rowmean(...)

stata_rowmiss(...)

stata_rownonmiss(..., strok = FALSE)
```

Arguments

`...`	Vectors and/or data frames: the variables of the row operation.
`missing`	For `stata_rowtotal()`: if `TRUE`, rows in which every value is missing get `NA` instead of 0 (Stata's `missing` option).
`strok`	For `stata_rownonmiss()`: allow string variables (Stata's `strok` option).

Value

`stata_rowtotal()` and `stata_rowmean()` return a double vector. `stata_rowmiss()` and `stata_rownonmiss()` return an integer vector. Every result has one element per row.

Semantics

The table and the rules below follow Stata manual [D] egen, except where noted. Rejecting dates is a statacompat design choice.

Row	rowtotal	rowtotal(missing = TRUE)	rowmean	rowmiss	rownonmiss
1, 2, 3	6	6	2	0	3
1, NA, 3	4	4	2	1	2
NA, NA, NA	**0**	NA	NA	3	0

- `stata_rowtotal()` treats missing values as 0, so a row with no non-missing values totals **0** by default. `missing = TRUE` corresponds to Stata's `missing` option and returns `NA` for such rows.
- `stata_rowmean()` ignores missing values and returns `NA` (not `NaN`) when every value in the row is missing.
- `stata_rowmiss()` accepts strings without any option and counts `""` as missing. [D] egen restricts string variables only for `rownonmiss()` (through `strok`) and states no such restriction for `rowmiss()`; this behaviour is inferred from that.
- `stata_rownonmiss()` rejects strings unless `strok = TRUE`, matching Stata's `strok` option; `""` then counts as missing.
- `stata_rowtotal()` and `stata_rowmean()` reject strings, as Stata does (type mismatch). Dates are also rejected, since summing dates is rarely meaningful; use `as.numeric()` if you really need it.
- Missing means `NA` or `NaN` (including `haven::tagged_na()`), and for strings `""` or `NA_character_`.

Precision

Results are doubles. Stata's default storage type for new variables is `float` (see `set type`), and `egen` without an explicit type is assumed to follow it. A `float` holds integers exactly only up to 16,777,216, so 16,777,217 cannot be stored and becomes 16,777,216. `stata_rowtotal()` gives 16,777,217, which corresponds to `egen double t = rowtotal(a b)`. Totals of Rupiah amounts and other large values can therefore differ from Stata output created without `double`. statacompat does not emulate `float` storage.

Inputs

Pass the variables either as separate vectors or as data frames, whose columns are used as variables, or a mixture. A data frame is never treated as a single vector. Inside `dplyr::mutate()`, use `dplyr::pick()` to select columns, e.g. `stata_rowtotal(dplyr::pick(r1:r5))`. All variables must have the same length; scalars are not recycled. Matrices are rejected; convert them with `as.data.frame()`.

Examples

```
r1 <- c(1, 1, NA)
r2 <- c(2, NA, NA)
r3 <- c(3, 3, NA)

stata_rowtotal(r1, r2, r3)
stata_rowtotal(r1, r2, r3, missing = TRUE)
stata_rowmean(r1, r2, r3)
stata_rowmiss(r1, r2, r3)
stata_rownonmiss(r1, r2, r3)

# base R gives NaN for the all-missing row
rowMeans(cbind(r1, r2, r3), na.rm = TRUE)

# data frames and strings
df <- data.frame(x = c(1, NA), s = c("a", ""))
```

```
stata_rowmiss(df)
stata_rownonmiss(df, strok = TRUE)

if (requireNamespace("dplyr", quietly = TRUE)) {
  dta <- data.frame(r1 = r1, r2 = r2, r3 = r3)
  dplyr::mutate(dta, total = stata_rowtotal(dplyr::pick(r1:r3)))
}
```

Index

as.data.frame(), *10*
as.integer(), *2*

ifelse(), *4*

stata_comparison, 2, *6, 7*
stata_cond, 4
stata_eq *(*stata_comparison*)*, 2
stata_eq(), *5, 6*
stata_ge *(*stata_comparison*)*, 2
stata_gt *(*stata_comparison*)*, 2
stata_inlist, 5
stata_inlist(), *7*
stata_inrange, 7
stata_inrange(), *3*, *6*
stata_le *(*stata_comparison*)*, 2
stata_lt *(*stata_comparison*)*, 2
stata_missing, 8
stata_missing(), *3*
stata_ne *(*stata_comparison*)*, 2
stata_row, 9
stata_rowmean *(*stata_row*)*, 9
stata_rowmiss *(*stata_row*)*, 9
stata_rowmiss(), *8*
stata_rownonmiss *(*stata_row*)*, 9
stata_rowtotal *(*stata_row*)*, 9