

# Package ‘variables’

May 8, 2026

**Title** Variable Descriptions

**Version** 1.1-2

**Date** 2025-05-09

**Description** Abstract descriptions of (yet) unobserved variables.

**URL** <http://ctm.R-forge.R-project.org>

**Imports** stats

**License** GPL-2

**NeedsCompilation** no

**Author** Torsten Hothorn [aut, cre] (ORCID:  
<<https://orcid.org/0000-0001-8301-0471>>)

**Maintainer** Torsten Hothorn <[Torsten.Hothorn@R-project.org](mailto:Torsten.Hothorn@R-project.org)>

**Repository** CRAN

**Date/Publication** 2025-05-09 15:00:02 UTC

## Contents

variables-package . . . . .	2
access . . . . .	2
check . . . . .	3
factor_var . . . . .	3
mkgrid . . . . .	4
numeric_var . . . . .	4
ordered_var . . . . .	5
vars . . . . .	6
<b>Index</b>	<b>8</b>

---

variables-package      *General Information on the **variables** Package*

---

### Description

The **variables** package offers a small collection of objects describing conceptual variables and corresponding methods, for example for generating a grid of values for a (yet) unmeasured variable.

The package was written to support the **basefun** and **mlt** packages and will be of limited use outside these packages.

### Author(s)

This package is authored by Torsten Hothorn <Torsten.Hothorn@R-project.org>.

### References

Torsten Hothorn (2018), Most Likely Transformations: The mlt Package, *Journal of Statistical Software*, forthcoming. URL: <https://cran.r-project.org/package=mlt.docreg>

---

access      *Accessor Functions*

---

### Description

Access properties of variable objects

### Usage

```
## S3 method for class 'var'
variable.names(object, ...)
desc(object)
unit(object)
support(object)
bounds(object)
is.bounded(object)
```

### Arguments

object	a variable object
...	additional arguments, currently not used

### Details

These generics have corresponding methods for [factor\\_var](#), [ordered\\_var](#) and [numeric\\_var](#) objects as well as for vars collections of those.

---

check	<i>Check Observations Against Formal Description</i>
-------	--

---

**Description**

Check if observations correspond to their formal descriptions

**Usage**

```
check(object, data)
```

**Arguments**

object	an object of class var or vars
data	a data.frame

**Details**

The function returns true if data matches the description in object.

---

factor_var	<i>Unordered Categorical Variable</i>
------------	---------------------------------------

---

**Description**

Formal description of an unordered categorical variable

**Usage**

```
factor_var(name, desc = NULL, levels, ...)
```

**Arguments**

name	character, the name of the variable
desc	character, a description of what is measured
levels	character, the levels of the factor
...	ignored

**Details**

A conceptual description of a (yet) unobserved unordered categorical variable.

**Value**

An object of class factor\\_var inheriting from var with corresponding methods.

**Examples**

```
factor_var("eye", "eye color", c("blue", "brown", "green", "grey", "mixed"))
```

---

mkgrid

*Generate Grids of Observations*


---

**Description**

Make a grid of values

**Usage**

```
mkgrid(object, ...)
## S3 method for class 'continuous_var'
mkgrid(object, n = 2, add = TRUE, ...)
```

**Arguments**

object	an object of class var or vars
n	number of grid points for a continuous variable
add	logical, adds the add argument (in <a href="#">numeric_var</a> ) to support if TRUE
...	additional arguments

**Details**

The function returns a names list of values for each variable.

---

numeric\_var

*Numeric Variable*


---

**Description**

Formal description of numeric variable

**Usage**

```
numeric_var(name, desc = NULL, unit = NULL, support = c(0, 1), add = c(0, 0),
            bounds = NULL, ...)
```

**Arguments**

name	character, the name of the variable
desc	character, a description of what is measured
unit	character, the measurement unit
support	the support of the measurements, see below
add	add these values to the support before generating a grid via <a href="#">mkgrid</a>
bounds	an interval defining the bounds of a real sample space
...	ignored

**Details**

A numeric variable can be discrete (support is then the set of all possible values, either integer or double; integers of length 2 are interpreted as all integers inbetween) or continuous (support is a double of length 2 giving the support of the data).

If a continuous variable is bounded, bounds defines the corresponding interval.

**Value**

An object of class `numeric\_var` inheriting from `var` with corresponding methods.

**Examples**

```
numeric_var("age", "age of patient", "years", support = 25:75)
numeric_var("time", "survival time", "days", support = 0:365)
numeric_var("time", "survival time", "days", support = c(0.0, 365),
            bounds = c(0, Inf))
```

---

ordered_var	<i>Ordered Categorical Variable</i>
-------------	-------------------------------------

---

**Description**

Formal description of an ordered categorical variable

**Usage**

```
ordered_var(name, desc = NULL, levels, sparse = FALSE, ...)
```

**Arguments**

name	character, the name of the variable
desc	character, a description of what is measured
levels	character, the ordered levels of the factor
sparse	logical, set-up a sparse model matrix
...	ignored

**Details**

A conceptual description of a (yet) unobserved ordered categorical variable.

**Value**

An object of class `ordered\_var` inheriting from `var` with corresponding methods.

**Examples**

```
ordered_var("temp", "temperature", c("cold", "lukewarm", "warm", "hot"))
```

---

vars

---

*Multiple Abstract Descriptions*


---

**Description**

Concatenate or generate multiple variable descriptions

**Usage**

```
## S3 method for class 'var'
c(...)
as.vars(object)
```

**Arguments**

object	an object
...	a list of variable objects

**Details**

`c()` can be used to concatenate multiple variable objects; the corresponding generics also work for the resulting object. `as.vars()` tries to infer a formal description from data.

**Examples**

```
f <- factor_var("x", levels = LETTERS[1:3])  
n <- numeric_var("y")
```

```
fn <- c(f, n)  
variable.names(fn)  
support(fn)  
is.bounded(fn)  
mkgrid(fn, n = 9)
```

```
as.vars(iris)
```

# Index

- \* **package**
  - variables-package, 2
- access, 2
- as.vars (vars), 6
- bounds (access), 2
- c.var (vars), 6
- check, 3
- desc (access), 2
- factor\_var, 2, 3
- is.bounded (access), 2
- mkgrid, 4, 5
- numeric\_var, 2, 4, 4
- ordered\_var, 2, 5
- support (access), 2
- unit (access), 2
- variable.names.var (access), 2
- variables (variables-package), 2
- variables-package, 2
- vars, 6