Package ‘spec’

May 9, 2026

Type Package

Title A Data Specification Format and Interface

Version 0.1.9

Author Tim Bergsma

Maintainer Tim Bergsma <`bergsmat@gmail.com`>

Description Creates a data specification that describes the columns of a table (data.frame). Provides methods to read, write, and update the specification. Checks whether a table matches its specification. See specification.data.frame(),read.spec(), write.spec(), as.csv.spec(), respecify.character(), and %matches%.data.frame().

Imports encode, csv, magrittr, utils

Depends R (>= 2.10)

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

NeedsCompilation no

Repository CRAN

Date/Publication 2024-02-18 05:30:07 UTC

Contents

as.spec 2
as.spec.character 2
as.spec.data.frame 3
drug 4
read.spec 5
respecify.character 5
respecify.spec 6
specification.data.frame 7
specify.character 8

specify.data.frame 8
write.spec 9
%matches%.character 10
%matches%.data.frame 10

Index **12**

`as.spec` *Coerce to Spec*

Description

Coerces to class spec, a specification object

Usage

```
as.spec(x, ...)
```

Arguments

`x`	object
`...`	passed arguments

See Also

Other as.spec: `as.spec.character()`, `as.spec.data.frame()`, `read.spec()`, `write.spec()`

`as.spec.character` *Coerce to Specification from Character*

Description

Coerces to specification from character (length-one filepath).

Usage

```
## S3 method for class 'character'
as.spec(x, ...)
```

Arguments

`x`	character path to spec-formatted file
`...`	passed arguments

Value

spec

See Also

Other as.spec: `as.spec.data.frame()`, `as.spec()`, `read.spec()`, `write.spec()`

Examples

```
data(drug)
file <- tempfile()
spec <- specification(drug, tol = 3)
write.spec(spec, file = file)
as.spec(file)
```

`as.spec.data.frame` *Coerce to Spec from Data Frame*

Description

Coerces to spec from data.frame already having basic properties.

Usage

```
## S3 method for class 'data.frame'
as.spec(x, ...)
```

Arguments

`x`	data.frame
`...`	passed arguments

Value

spec

See Also

Other as.spec: `as.spec.character()`, `as.spec()`, `read.spec()`, `write.spec()`

Examples

```
data(drug)
as.spec(specification(drug, tol = 3))
```

`drug` *Simulated Pharmacometric Data*

Description

A fictitious dataset giving doses and pharmacometric samples for multiple subjects in an imaginary Phase * drug trial.

Usage

```
drug
```

Format

A data frame with 600 rows and 24 variables:

C a comment flag, typically NA but 'C' for records that should be ignored

ID integer subject identifier

TIME relative time (h)

SEQ sequence identifier to break ties when sorting

EVID event type identifier, 0: pk sample, 1: dose

AMT drug amount (mg)

DV plasma drug concentration (ng/mL)

SUBJ subject identifier

HOUR nominal hour (h)

HEIGHT height (cm)

WEIGHT weight (kg)

SEX sex, 0: female, 1: male

AGE age (y)

DOSE dose group (mg)

FED prandial state, 0: fasted, 1: fed

SMK smoker status, 0: non, 1: smoker

DS disease state, 0: no disease

CRCN normalized creatinine clearance (mL/min)

TAFD time since first dose (h)

TAD time since most recent dose (h)

LDOS amount of most recent dose (mg)

MDV missing dependent value, 0: not missing, 1: missing

predose predose flag, 0: record not predose, 1: record is predose

zerodv zero DV flag, 0, DV not zero, 1: DV is zero

`read.spec` *Read Specification from File*

Description

Reads specification from file. If first line contains tab characters, assumes format is tab-delimited text. Otherwise, assumes format is comma-separated variable (csv).

Usage

```
read.spec(x, clean = TRUE, ...)
```

Arguments

`x`	character (file path)
`clean`	whether to strip balanced double quotes and outer white space from character values
`...`	passed arguments (ignored)

Value

spec

See Also

Other as.spec: `as.spec.character()`, `as.spec.data.frame()`, `as.spec()`, `write.spec()`

Examples

```
data(drug)
file <- tempfile()
spec <- specification(drug, tol = 3)
write.spec(spec, file = file)
read.spec(file)
```

`respecify.character` *Respecify Character*

Description

Respecify specification, supplied as filepath. Updates numeric ranges. Useful if these have changed and spec no longer matches.

Usage

```
## S3 method for class 'character'
respecify(x, data = sub("spec$", "csv", x), file = x, ...)
```

Arguments

`x`	character filepath for a spec file (*.spec)
`data`	character filepath for a dataset
`file`	where to write the result (over-write source, by default)
`...`	passed arguments

See Also

Other respecify: `respecify.spec()`, `respecify()`

`respecify.spec` *Respecify Specification*

Description

Respecify specification. Updates numeric ranges. Useful if these have changed and spec no longer matches.

Usage

```
## S3 method for class 'spec'
respecify(x, data, file = NULL, ...)
```

Arguments

`x`	spec
`data`	a data.frame or path to csv file
`file`	where to write the result (default: do not write)
`...`	passed arguments

See Also

Other respecify: `respecify.character()`, `respecify()`

Examples

```
data(drug)
file <- tempfile()
spec <- specification(drug,tol = 3)
write.spec(spec, file = file)
drug %matches% spec
drug %matches% file
max <- max(drug$DV,na.rm=TRUE)
drug$DV[!is.na(drug$DV) & drug$DV == max] <- max + 1
drug %matches% file
respecify(file, drug)
drug %matches% file
```

`specification.data.frame`

Make a Specification for a Data Frame

Description

Makes a specification for data.frame. Creates a template based on the data.frame. Uses column names for labels where columns do not have a label attribute. Factors will be encoded. numerics will be rounded to `digits` and like integers will be expressed as ranges in `guide` column. Integers and character with less than or exactly `tol` unique values will be encoded.

Usage

```
## S3 method for class 'data.frame'
specification(x, tol = 10, digits = 20, ...)
```

Arguments

`x`	object
`tol`	integer
`digits`	integer
`...`	passed arguments

Value

spec data.frame with columns as follows.

column Column name.

label A descriptive label. Save and edit as necessary using external tool.

guide A guide to interpretation. NA for arbitrary character; range [low:high] for integer and numeric; an encoding e.g. //0/no//1/yes// for factor-like items ... save and edit factor labels as necessary using external tool.

For numeric ranges you can add text, such as units. E.g. if default guide is ’[0:100]’ you can edit to give ’mg [0:100]’. Or you can just substitute ’mg’. `guidetext` extracts just the character portion, and `matches` enforces the numeric range.

required An R expression that can be coerced to logical. TRUE means item cannot be NA.

comment Arbitrary comment, e.g. derivation of the item given by `column`.

See Also

`link{read.spec}` `write.spec` `respecify.character` `write.spec` `matches`

Other specification: `specification.comment()`, `specification.default()`, `specification()`

Examples

```
data(drug)
file <- tempfile()
spec <- specification(drug, tol = 3)
```

`specify.character` *Specify Character*

Description

Attach specifics to a data.frame, supplied as csv filepath.

Usage

```
## S3 method for class 'character'
specify(x, file = sub("csv$", "spec", x), spec = read.spec(file), ...)
```

Arguments

`x`	character filepath for a csv file
`file`	character filepath for a matching spec file (ignored if spec provided)
`spec`	a data specification (spec)
`...`	passed arguments

See Also

Other specify: `specify.data.frame()`, `specify()`

`specify.data.frame` *Specify Data Frame*

Description

Attach specifics to a data.frame as attributes, including label and guide.

Usage

```
## S3 method for class 'data.frame'
specify(x, spec, na.rm = TRUE, empty.rm = TRUE, ...)
```

Arguments

`x`	data.frame
`spec`	a data spec (or corresponding filepath) to use as source of attributes
`na.rm`	if TRUE, don't assign NA where encountered
`empty.rm`	if TRUE, don't assign empty string where encountered
`...`	passed arguments

See Also

Other specify: `specify.character()`, `specify()`

Examples

```
data(drug)
spec <- specification(drug,tol = 3)
drug %matches% spec
drug <- specify(drug,spec)
attributes(drug$HEIGHT)
```

`write.spec` *Write Specification to Storage*

Description

Writes specification to storage in tab-delimited format. Use as.csv() for CSV format.

Usage

```
write.spec(x, file, ...)
```

Arguments

`x`	spec
`file`	character filepath for storage location
`...`	passed arguments

See Also

Other as.spec: `as.spec.character()`, `as.spec.data.frame()`, `as.spec()`, `read.spec()`

Examples

```
data(drug)
file <- tempfile()
spec <- specification(drug, tol = 3)
write.spec(spec, file = file)
```

`%matches%.character` *Check Whether Character matches y*

Description

Checks whether character matches y, treating x as filepath.

Usage

```
## S3 method for class 'character'
x %matches% y, ...
```

Arguments

`x`	character
`y`	object
`...`	passed arguments

See Also

Other matches: `%matches%.data.frame`(), `%matches%.spec`(), `%matches%`()

Examples

```
data(drug)
file <- tempfile()
spec <- specification(drug, tol = 3)
library(csv)
as.csv(drug, file)
file %matches% spec
```

`%matches%.data.frame` *Check Whether Data Frame matches Spec*

Description

Checks whether data.frame matches spec. Column names, count, and order are enforced. Encodings are enforced (all non-missing values must be valid codes). Integer and numeric ranges are enforced. Values of `required` are parsed and evaluated in data context: Where TRUE, the corresponding data value for `column` cannot be missing.

Usage

```
## S3 method for class 'data.frame'
x %matches% y, ...
```

Arguments

`x`	spec
`y`	coerced to spec (spec object or filepath for spec file).
`...`	passed arguments

Value

logical; TRUE if all checks above are enforceable.

See Also

Other matches: `%matches%.character()`, `%matches%.spec()`, `%matches%()`

Examples

```
data(drug)
file <- tempfile()
spec <- specification(drug, tol = 3)
write.spec(spec, file = file)
drug %matches% spec
```

Index

∗ **as.spec**
    `as.spec`, 2
    `as.spec.character`, 2
    `as.spec.data.frame`, 3
    `read.spec`, 5
    `write.spec`, 9
∗ **datasets**
    `drug`, 4
∗ **matches**
    `%matches%.character`, 10
    `%matches%.data.frame`, 10
∗ **respecify**
    `respecify.character`, 5
    `respecify.spec`, 6
∗ **specification**
    `specification.data.frame`, 7
∗ **specify**
    `specify.character`, 8
    `specify.data.frame`, 8
`%matches%`, *10, 11*
`%matches%.character`, 10, *11*
`%matches%.data.frame`, *10*, 10
`%matches%.spec`, *10, 11*

`as.spec`, 2, *3*, *5*, *9*
`as.spec.character`, *2*, 2, *3*, *5*, *9*
`as.spec.data.frame`, *2, 3*, 3, *5*, *9*

`drug`, 4

`guidetext`, *7*

`matches`, *7*
`matches` *(*`%matches%.data.frame`*)*, 10

`read.spec`, *2, 3*, 5, *9*
`respecify`, *6*
`respecify.character`, 5, *6, 7*
`respecify.spec`, *6*, 6

`spec` *(*`specification.data.frame`*)*, 7
`specification`, *7*
`specification.comment`, *7*
`specification.data.frame`, 7
`specification.default`, *7*
`specify`, *8, 9*
`specify.character`, 8, *9*
`specify.data.frame`, *8*, 8

`write.spec`, *2, 3*, *5*, *7*, 9