Package 'newdata'

November 7, 2026

Title Generate New Data Frames for Prediction

Version 0.1.0

Description Generates new data frames for predictive purposes. By default, all specified variables vary across their range while all other variables are held constant at the default reference value. Types, classes, factor levels and time zones are always preserved. The user can specify the length of each sequence, require that only observed values and combinations are used and add new variables. The result is intended to be passed to the prediction method of a fitted model object to visualize the effect of one or more variables while controlling for the others.

License MIT + file LICENSE

URL https://poissonconsulting.github.io/newdata/, https://github.com/poissonconsulting/newdata/

BugReports https://github.com/poissonconsulting/newdata/issues

Depends R (>= 4.1)

Imports chk, dplyr, hms, lifecycle, rlang, tibble, tidyr, vctrs

Suggests covr, datasets, spelling, testthat (>= 3.0.0), withr

Config/Needs/website poissonconsulting/poissontemplate

Config/roxygen2/version 8.1.0.9000

Config/testthat/edition 3

Encoding UTF-8

Language en-US

LazyData true

NeedsCompilation no

Author Joe Thorley [aut, cre] (ORCID: <https://orcid.org/0000-0002-7683-4592>), Kirill Müller [aut] (ORCID: <https://orcid.org/0000-0002-1416-3412>), Ayla Pearson [aut] (ORCID: <https://orcid.org/0000-0001-7388-1222>), Nadine Hussein [ctb] (ORCID: <https://orcid.org/0000-0003-4470-8361>), Maëlle Salmon [ctb] (ORCID: <https://orcid.org/0000-0002-2815-0399>), Poisson Consulting [fnd, cph]

Maintainer Joe Thorley <joe@poissonconsulting.ca>

Repository CRAN

Date/Publication 2026-11-07 14:30:08 UTC

Contents

new_data 2
new_seq 3
new_value 8
old_data 9
xcast 10
xnew_data 11
xnew_seq 12
xnew_value 13
xobs_only 14

Index **16**

`new_data` *Generate New Data* **[Superseded]**

Description

Generates a new data frame (in the form of a tibble) with each variable held constant or varying as a unique ordered sequence.

Usage

```
new_data(
  data,
  seq = character(0),
  ref = list(),
  obs_only = list(character(0)),
  length_out = 30
)
```

Arguments

`data`	The data frame to generate the new data from.
`seq`	A character vector of the names of the variables in `data` to generate sequences for. All other variables are held constant at their reference value.
`ref`	**[Deprecated]** A named list of reference values for variables that are not in `seq`. Using it produces a warning. Instead name the value in `xnew_data()`, for example `xnew_data(data, a, b = 1)`.

obs_only **[Deprecated]** A list of character vectors indicating the sets of variables to only allow observed combinations for. If TRUE then obs_only is set to be seq. Using it produces a warning. Instead use xobs_only() in xnew_data(), for example xnew_data(data, xobs_only(a, b)).

length_out A count indicating the maximum length of sequences for all types of variables except logical, character, factor and ordered factors.

Details

new_data() has been superseded by xnew_data(), which accepts bare column names and provides finer control through xnew_seq(), xnew_value(), xobs_only() and xcast(). new_data() remains fully functional and is maintained for backwards compatibility with existing code. All of its arguments continue to work. The ref and obs_only arguments are deprecated, which means that using them produces a warning but they still behave as documented.

The call new_data(data, seq = c("a", "b"), length_out = 30) is equivalent to xnew_data(data, a, b, .length_out = 30).

Value

A tibble of the new data.

See Also

xnew_data()

Examples

```
new_data(old_data, "int")
new_data(old_data, "dbl")
new_data(old_data, c("int", "dbl"))

# equivalent using xnew_data()
xnew_data(old_data, int)
xnew_data(old_data, dbl)
xnew_data(old_data, int, dbl)
```

---

new_seq *Generate New Sequence*

Description

Generate a new sequence of values. A sequence of values is used to predict the effect of a variable.

Usage

```
new_seq(
  x,
  .length_out = NULL,
  ...,
  .obs_only = NULL,
  length_out = deprecated(),
  obs_only = deprecated()
)

## Default S3 method:
new_seq(x, .length_out = NULL, ..., .obs_only = NULL)

## S3 method for class 'logical'
new_seq(
  x,
  .length_out = NULL,
  ...,
  .obs_only = NULL,
  length_out = deprecated(),
  obs_only = deprecated()
)

## S3 method for class 'integer'
new_seq(
  x,
  .length_out = NULL,
  ...,
  .obs_only = NULL,
  length_out = deprecated(),
  obs_only = deprecated()
)

## S3 method for class 'double'
new_seq(
  x,
  .length_out = NULL,
  ...,
  .obs_only = NULL,
  length_out = deprecated(),
  obs_only = deprecated()
)

## S3 method for class 'character'
new_seq(
  x,
  .length_out = NULL,
  ...,
```

```
  .obs_only = NULL,
  length_out = deprecated(),
  obs_only = deprecated()
)

## S3 method for class 'factor'
new_seq(
  x,
  .length_out = NULL,
  ...,
  .obs_only = NULL,
  length_out = deprecated(),
  obs_only = deprecated()
)

## S3 method for class 'ordered'
new_seq(
  x,
  .length_out = NULL,
  ...,
  .obs_only = NULL,
  length_out = deprecated(),
  obs_only = deprecated()
)

## S3 method for class 'Date'
new_seq(
  x,
  .length_out = NULL,
  ...,
  .obs_only = NULL,
  length_out = deprecated(),
  obs_only = deprecated()
)

## S3 method for class 'POSIXct'
new_seq(
  x,
  .length_out = NULL,
  ...,
  .obs_only = NULL,
  length_out = deprecated(),
  obs_only = deprecated()
)

## S3 method for class 'hms'
new_seq(
  x,
```

```
  .length_out = NULL,
  ...,
  .obs_only = NULL,
  length_out = deprecated(),
  obs_only = deprecated()
)
```

Arguments

x	The object to generate the sequence from.
.length_out	The maximum length of the sequence.
...	These dots are for future extensions and must be empty.
.obs_only	A flag specifying whether to only use observed values.
length_out	**[Deprecated]** Use .length_out instead.
obs_only	**[Deprecated]** Use .obs_only instead.

Details

By default the sequence of values for objects of class numeric is 30 evenly space values across the range of the data. Missing values are always removed unless it's the only value or the object is zero length. The length of the sequence can be varied using the .length_out argument which gives the reference value when 1 and can even be 0. For integer objects the sequence is the unique integers. For character objects it's the actual values sorted by how common they are followed by their actual value. For factors it's the factor levels in order with the trailing levels dropped first. For ordered factors the intermediate levels are dropped first. For Date vectors it's the unique dates; same for hms vectors. For POSIXct vectors the time zone is preserved. For logical objects the longest possible sequence is c(FALSE, TRUE) so values of .length_out greater than 2 are equivalent to 2.

Value

A vector of the same class as the object.

Methods (by class)

- new_seq(default): Informative error for unsupported objects such as data frames and lists
- new_seq(logical): Generate new sequence of values for logical objects
- new_seq(integer): Generate new sequence of values for integer objects
- new_seq(double): Generate new sequence of values for double objects
- new_seq(character): Generate new sequence of values for character objects
- new_seq(factor): Generate new sequence of values for factors
- new_seq(ordered): Generate new sequence of values for ordered factors
- new_seq(Date): Generate new sequence of values for Date vectors
- new_seq(POSIXct): Generate new sequence of values for POSIXct vectors
- new_seq(hms): Generate new sequence of values for hms vectors

See Also

new_value() and new_data().

Examples

```
# by default the sequence of values for objects of class numeric
# is 30 evenly space values across the range of the data
new_seq(c(1, 4))
# missing values are always removed
new_seq(c(1, 4, NA))
# unless it's the only value
new_seq(NA_real_)
# or the object is zero length
new_seq(numeric())
# the length of the sequence can be varied using the .length_out argument
new_seq(c(1, 4), .length_out = 3)
new_seq(c(1, 4), .length_out = 2)
# which gives the reference value when 1
new_seq(c(1, 4), .length_out = 1)
# and can even be 0
new_seq(c(1, 4), .length_out = 0)
# for integer objects the sequence is the unique integers
new_seq(c(1L, 4L))
new_seq(c(1L, 100L))
# for character objects it's the actual values sorted by
# how common they are followed by their actual value
new_seq(c("a", "c", "c", "b", "b"))
new_seq(c("a", "c", "c", "b", "b"), .length_out = 2)
# for factors its the factor levels in order
new_seq(factor(c("a", "b", "c", "c"), levels = c("b", "a", "g")))
# with the trailing levels dropped first
new_seq(factor(c("a", "b", "c", "c"), levels = c("b", "a", "g")),
  .length_out = 2
)
# for ordered factors the intermediate levels are dropped first
new_seq(ordered(c("a", "b", "c", "c"), levels = c("b", "a", "g")),
  .length_out = 2
)
# for Date vectors it's the unique dates
new_seq(as.Date(c("2000-01-01", "2000-01-04")))
# same for hms vectors
new_seq(hms::as_hms(c("00:00:01", "00:00:04")))
# for POSIXct vectors the time zone is preserved
new_seq(as.POSIXct(c("2000-01-01 00:00:01", "2000-01-01 00:00:04"),
  tz = "PST8PDT"
))
# for logical objects the longest possible sequence is `c(FALSE, TRUE)`
new_seq(c(TRUE, TRUE, FALSE), .length_out = 3)
```

new_value *Generate New Reference Value*

Description

Generate a new reference value for a vector.

Usage

```
new_value(x, ..., .obs_only = NULL, obs_only = deprecated())
```

Arguments

x	The object to generate the reference value from.
...	These dots are for future extensions and must be empty.
.obs_only, obs_only	A flag specifying whether to only use observed values.

Details

By default the reference value for double vectors is the mean, unless .obs_only = TRUE, in which case its the median of the unique values. For integer vectors it's the floored mean unless .obs_only = TRUE, in which case it's also the median of the unique values. For character vectors it's the minimum of the most common values while for factors it's the first level. Ordered factors, Dates, times (hms), POSIXct and logical vectors are treated like integers. The factor levels and time zone are preserved.

Value

A scalar of the same class as the object.

See Also

xnew_value() and new_seq().

Examples

```
# the reference value for objects of class numeric is the mean
new_value(c(1, 4))
# unless .obs_only = TRUE, in which case its the median of the unique values
new_value(c(1, 4), .obs_only = TRUE)

# for integer objects it's the floored mean
new_value(c(1L, 4L))

# for character objects it's the minimum of the most common values
new_value(c("a", "b", "c", "c", "b"))
```

```
# for factors its the first level and the factor levels are preserved
new_value(factor(c("a", "b", "c", "c"), levels = c("b", "a", "g")))

# other classes are treated like integers
new_value(ordered(c("a", "b", "c", "c"), levels = c("b", "a", "g")))
new_value(as.Date(c("2000-01-01", "2000-01-04")))
new_value(hms::as_hms(c("00:00:01", "00:00:04")))
new_value(as.POSIXct(c("2000-01-01 00:00:01", "2000-01-01 00:00:04"),
  tz = "PST8PDT"
))
new_value(c(TRUE, FALSE, TRUE))
```

`old_data` *Example Data*

Description

An example tibble of example 'old' data.

Usage

```
old_data
```

Format

A tibble with 3 rows and 9 columns:

lgl A logical vector.

int An integer vector.

dbl A double vector.

chr A character vector.

fct A factor.

ord An ordered factor.

dte A Date vector.

dtt A POSIXct vector.

hms A hms vector.

Examples

```
old_data
```

xcast *Cast New Values for* xnew_data()

Description

Casts a sequence of values to the same class as the original vector.

Usage

```
xcast(..., .data = xnew_data_env$data)
```

Arguments

...	Named vectors of values to cast to the class of the identically named columns in .data.
.data	Normally defined by xnew_data(), users must pass a data frame or tibble if using this function directly.

Details

xcast() is a wrapper function on vctrs::vec_cast() for use in xnew_data() to avoid having to repeating the column name.

The casts that are possible are those defined by vctrs and the packages that provide the classes. For example, numeric and character values can be cast to hms because the hms package provides those casts, but vctrs does not provide casts from numeric or character values to Date or POSIXct as they require an origin, format or time zone. Date and POSIXct values must therefore be created with the required class, for example using as.Date() or as.POSIXct(). POSIXct values are converted to the time zone of the column.

Value

A tibble of the cast values.

See Also

vctrs::vec_cast() and xnew_data()

Examples

```
data <- tibble::tibble(
  period = factor(c("before", "before", "after", "after"),
    levels = c("before", "after")
  ),
  annual = factor(c(1, 3, 5, 8), levels = c(1, 3, 5, 8))
)

xnew_data(data, xcast(period = "before"))
xnew_data(data, xcast(period = "before", annual = c("1", "3")))
```

```
# numeric and character values can be cast to hms
xnew_data(old_data, xcast(hms = "01:01:11"))
# but Date and POSIXct values must be created with the required class
xnew_data(
  old_data,
  xcast(
    dte = as.Date("2024-01-01"),
    dtt = as.POSIXct("2024-01-01 10:30:42", tz = "PST8PDT")
  )
)
```

`xnew_data` *Generate New Data by Expansion*

Description

Generates a new data frame (in the form of a tibble)

Usage

```
xnew_data(.data, ..., .length_out = NULL)
```

Arguments

`.data`	The data frame to generate the new data from.
`...`	A list of variables to generate sequences for.
`.length_out`	NULL or a count specifying the maximum length of all sequences.

Details

By default, all specified variables vary across their range while all other variables are held constant at their reference value. Types, classes, factor levels and time zones are always preserved. The user can specify the length of each sequence, require that only observed values and combinations are used and add new variables.

Naming an argument generates a new column of that name rather than varying an existing column. A bare vector or a one column data frame becomes the new column, so `Annual = annual` keeps `annual` at its reference value and adds `Annual` varying across its range. A data frame with more than one column, such as the result of `tidyr::nesting()`, is packed into a data frame column of that name. An argument that evaluates to `NULL` is dropped.

Value

A tibble of the new data.

See Also

`xnew_value()`, `xnew_seq()`, `xcast()` and `xobs_only()`

Examples

```
data <- tibble::tibble(
  period = factor(c("before", "before", "after", "after"),
    levels = c("before", "after")
  ),
  count = c(0L, 1L, 5L, 4L),
  annual = factor(c(2, 3, 5, 8), levels = c(1, 2, 3, 5, 8))
)

# By default all other variables are held constant at their reference value.
xnew_data(data)

# Specifying a variable causes it to vary across its range.
xnew_data(data, annual)

# The user can specify the length of a sequence.
xnew_data(data, xnew_seq(annual, .length_out = 3))

# And only allow observed values.
xnew_data(data, xnew_seq(annual, .length_out = 3, .obs_only = TRUE))

# With multiple variables all combinations are produced
xnew_data(data, period, xnew_seq(annual, .length_out = 3, .obs_only = TRUE))

# Naming a variable generates a new column of that name
xnew_data(data, Annual = annual)

# The new variable can be generated using an external vector, too
new_annual <- unique(data$annual)
xnew_data(data, Annual = new_annual)

# To only preserve observed combinations use
xnew_data(data, xobs_only(period, annual))

# And to cast the values use
xnew_data(data, xcast(annual = "3"))
```

---

xnew_seq *Generate New Sequence for* xnew_data()

Description

Generate a new sequence of values for a vector.

Usage

```
xnew_seq(x, ...)
```

Arguments

x	The object to generate the sequence from.
...	Additional arguments passed to new_seq().

Details

xnew_seq() is a wrapper function on new_seq() for use in xnew_data() to avoid having to repeating the column name.

Value

A one column tibble named after x of the new sequence.

See Also

new_seq() and xnew_data().

Examples

```
data <- tibble::tibble(
  a = c(1L, 3L, 4L),
  b = c(4, 4.5, 6),
  d = c("a", "b", "c")
)

xnew_data(data, a, b = new_seq(b, .length_out = 3), xnew_seq(d, .length_out = 2))
```

---

xnew_value *Generate New Reference Value for* xnew_data()

Description

Generate a new reference value for a vector.

Usage

```
xnew_value(x, ...)
```

Arguments

x	The object to generate the reference value from.
...	Additional arguments passed to new_value().

Details

xnew_value() is a wrapper function on new_value() for use in xnew_data() to avoid having to repeating the column name.

Value

A one row, one column tibble named after x of the new reference value.

See Also

new_value() and xnew_data().

Examples

```
data <- tibble::tibble(
  a = c(1L, 3L, 4L),
  b = c(4, 4.5, 6),
  d = c("a", "b", "c")
)

xnew_data(data, a, b = new_value(b), xnew_value(d))
```

---

xobs_only *Generate Observed Combinations Only*

Description

Generates the observed combinations of one or more variables.

Usage

```
xobs_only(..., .length_out = NULL, .data = xnew_data_env$data)
```

Arguments

...	One or more unnamed variables in .data to generate observed combinations for. Naming an argument is an error as a new column has no observed combinations to preserve; use a named argument to xnew_data() instead.
.length_out	A count to override the default length of sequences.
.data	Normally defined by xnew_data(), users must pass a data frame or tibble if using this function directly.

Details

By default xnew_data() generates all the combinations of the variables it is given. xobs_only() drops the combinations that do not occur in the data.

Value

A tibble of the observed combinations of the variables.

See Also

xnew_data()

Examples

```
data <- tibble::tibble(
  period = factor(c("before", "before", "after", "after"),
    levels = c("before", "after")
  ),
  annual = factor(c(1, 3, 5, 8), levels = c(1, 3, 5, 8))
)
xnew_data(data, period, annual)
xnew_data(data, xobs_only(period, annual))
xnew_data(data, xobs_only(period, xnew_seq(annual, .length_out = 3)))
```

Index

∗ **datasets**
old_data, 9

as.Date(), *10*
as.POSIXct(), *10*

new_data, 2
new_data(), *7*
new_seq, 3
new_seq(), *8*, *13*
new_value, 8
new_value(), *7*, *13, 14*

old_data, 9

tidyr::nesting(), *11*

vctrs::vec_cast(), *10*

xcast, 10
xcast(), *3*, *11*
xnew_data, 11
xnew_data(), *2, 3*, *10*, *13–15*
xnew_seq, 12
xnew_seq(), *3*, *11*
xnew_value, 13
xnew_value(), *3*, *8*, *11*
xobs_only, 14
xobs_only(), *3*, *11*