Package: R6Nomogram (via r-universe)

July 7, 2026

Version 1.0

Title Create, Edit, and Plot Nomograms using R6 Objects

License GPL (>= 2)

Maintainer Greg Snow <538280@gmail.com>

Description Nomograms are a type of plot for displaying linear models. A scale is plotted for each predictor in the model that translates values of the variable into ``points", the sum of the ``points" is then looked up on another scale to find the final prediction from the model. This package provides an R6 object constructor that does the computations for you to create an object representing the nomogram for the model. Methods and fields in the object allow you to customize the nomogram. You can then plot the nomogram, further customize, replot, etc. These types of nomograms are described in Harrell (2015) <doi:10.1007/978-3-319-19425-7>.

URL https://github.com/glsnow/R6Nomogram

Imports R6

Suggests quarto

Encoding UTF-8

Config/roxygen2/version 8.0.0

VignetteBuilder quarto

Repository https://glsnow.r-universe.dev

Date/Publication 2026-06-30 19:03:05 UTC

RemoteUrl https://github.com/glsnow/r6nomogram

RemoteRef HEAD

RemoteSha 6eacaa57795ac704196f9afeb20c87aee8aee151

RemoteSubdir pkg

Contents

R6Nomogram *Nomogram R6 Object*

Description

R6Nomogram builds and plots a nomogram from a fitted regression model (e.g. lm, glm, coxph, ...). Given the model and the data used to fit it, the object computes, for each predictor (or interaction term) in the model, an axis that maps predictor values to a common "points" scale. The points scale is in turn related back to the model's linear predictor and to the fitted response, so that the nomogram can be read by summing points across predictors and converting the total back to a predicted response.

Details

The construction of a nomogram proceeds through a small pipeline of steps, all of which are run automatically by initialize() by default (and can also be re-run individually if axes need to be adjusted):

1. new or initialize - initialize object and store model, terms, and newdat in the object.
2. PopulateVars() - extract variable names, term-level predictions, and response predictions from the model and data.
3. create.x() - build sorted, de-duplicated predictor values and the corresponding points for each axis.
4. shift() - shift each axis so that point values start at a convenient origin (by default, 0).
5. scale() - rescale all axes so that the maximum number of points is a convenient round number (by default, 100).
6. pretty() / pretty.y() - choose "pretty" tick locations for the predictor axes and for the total points / linear predictor / response axis.

Once built, plot() draws the nomogram, and tables() returns the same information as a list of data frames (e.g. for use outside of a plot).

Public fields

model The fitted model object (e.g. from lm, glm, coxph, etc.) used to construct the nomogram.

terms The terms object describing the model formula (generated by the predict method with type='terms').

newdata The data frame used to build the nomogram, either supplied by the user or extracted from model.

var.names Character vector of the variable names appearing in the model formula (excluding the response).

x.names Character vector of names for each predictor/interaction axis on the nomogram (interaction terms are joined with ":").

x.labels Character vector of axis labels used when plotting (defaults to x.names).

`orig.x` Data frame of the original predictor variable values extracted from `newdata`.

`orig.terms` Data frame of the model's term-level (partial) predictions, one column per term.

`orig.response` Vector of unique predicted response values (response scale).

`x.vals` List, one element per axis, of the unique, sorted predictor (or interaction) values.

`x.points` List, one element per axis, of the points corresponding to `x.vals`.

`x.pretty.vals` List of "pretty" axis tick values used for plotting.

`x.pretty.points` List of point positions corresponding to `x.pretty.vals`.

`x.y.offsets` List controlling the vertical offset/justification of axis labels when plotting.

`scale.c` Numeric scaling constant relating points back to the linear predictor scale.

`constant` The model intercept/offset constant, adjusted as points are shifted and scaled.

`total.points` Numeric vector of total points corresponding to the unique fitted values.

`linear.predictor` Numeric vector of linear predictor values corresponding to `total.points`.

`pretty.total.points` Numeric vector of "pretty" total-point tick values for the response axis.

`pretty.linear.predictor` Numeric vector of linear predictor values matching `pretty.total.points`.

`pretty.response` Numeric vector of response values matching `pretty.total.points`.

`plot.lp` Logical; whether to plot a separate linear-predictor axis in addition to total points and response.

`v.pos.x` Numeric vector of vertical plotting positions for the predictor axes.

`v.pos.r` Numeric vector of vertical plotting positions for the total points / linear predictor / response axes.

`options` List of plotting options: `tik.len`, `txt.pos`, `points.nint`, `signif.digits`, `text.par`, `line.par`, and `tick.par`. The last 3 default to empty lists, but can be lists with any of the parameters that can be passed to `text` or `segments`. They can also be a list of lists with the element names matching `self$x.names` to give different options for each scale.

`points.lab` Label for the "Points" axis.

`total.points.lab` Label for the "Total Points" axis.

`lp.lab` Label for the "Linear Predictor" axis.

`resp.lab` Label for the "Response" axis.

`tp.range` Numeric vector giving the range of total points used to position the response axis.

`verbose` Logical; whether to print progress messages.

`par` list of parameter settings in place during the last plot. This can be used with the `par` function if you are manually adding to the plot.

Methods

Public methods:

- `R6Nomogram$new()`
- `R6Nomogram$PopulateVars()`
- `R6Nomogram$create.x()`
- `R6Nomogram$shift()`

- R6Nomogram$scale()
- R6Nomogram$pretty()
- R6Nomogram$pretty.y()
- R6Nomogram$plot()
- R6Nomogram$grconvertX()
- R6Nomogram$tables()
- R6Nomogram$clone()

`R6Nomogram$new()`: Construct a new `R6Nomogram`. Runs the full construction pipeline (loading data, populating variables, creating axes, shifting, scaling, and prettying) up to the requested `steps`.

Usage:

```
R6Nomogram$new(
  model,
  newdata,
  verbose = TRUE,
  steps = Inf,
  type.terms = "terms",
  type.response = "response"
)
```

Arguments:

`model` A fitted model object (e.g. from `lm`, `glm`, or `coxph`).

`newdata` Optional data frame to use instead of the data stored in/with `model`. If missing, the data is recovered from `model$data`, the `data` argument of `model$call`, or `model.frame(model)`, in that order.

`verbose` Logical; print progress messages for each step.

`steps` Numeric; how many construction steps to run (useful for stopping partway through, e.g. to inspect or adjust axes by hand before continuing). Defaults to `Inf`, i.e. run all steps.

`type.terms` The `type` argument passed to `predict()` to obtain term-level predictions.

`type.response` The `type` argument passed to `predict()` to obtain response-scale predictions.

Returns: The object itself, invisibly.

`R6Nomogram$PopulateVars()`: Extract variable names, term-level predictions, and response predictions from `model` and `newdata`, and use them to populate `x.names`, `x.labels`, `orig.x`, `orig.terms`, `orig.response`, `constant`, `total.points`, and `linear.predictor`.

Usage:

```
R6Nomogram$PopulateVars(
  model = self$model,
  newdata = self$newdata,
  type.terms = "terms",
  type.response = "response"
)
```

Arguments:

`model` A fitted model object. Defaults to `self$model`.

`newdata` A data frame of predictor values. Defaults to `self$newdata`.

`type.terms` The `type` argument passed to `predict()` to obtain term-level predictions.

`type.response` The `type` argument passed to `predict()` to obtain response-scale predictions.

Returns: The object itself, invisibly.

`R6Nomogram$create.x()`**:** Build, for each axis (predictor or interaction term), the unique, sorted predictor values (`x.vals`) and their corresponding points (`x.points`), derived from `orig.x` and `orig.terms`.

Usage:

```
R6Nomogram$create.x()
```

Returns: The object itself, invisibly.

`R6Nomogram$shift()`**:** Shift the points for one or more axes by a constant offset. By default, each axis named in `w` is shifted so that its minimum point value becomes 0; `total.points` and `constant` are updated to match so that the overall model is unaffected.

Usage:

```
R6Nomogram$shift(w = self$x.names, v, update.constant = TRUE)
```

Arguments:

`w` Character vector of axis names (from `x.names`) to shift. Defaults to all axes.

`v` Numeric vector of shift amounts, one per element of `w` (matched by name if named). If missing, each axis is shifted so that the minimum number of points is 0.

`update.constant` Logical; if `TRUE` (the default), `total.points` and `constant` are updated so totals remain consistent with the shifted axes.

Returns: The object itself, invisibly.

`R6Nomogram$scale()`**:** Rescale all axes (and `total.points`) so that the largest point value across all axes equals `max.points`. `scale.c` is updated so that points can still be converted back to the linear predictor scale.

Usage:

```
R6Nomogram$scale(max.points = 100)
```

Arguments:

`max.points` The desired maximum number of points. Defaults to 100.

Returns: The object itself, invisibly.

`R6Nomogram$pretty()`**:** Choose "pretty" tick locations for one or more predictor axes, by interpolating (via a natural spline of points on values) the points corresponding to each chosen tick value. Results are stored in `x.pretty.vals` and `x.pretty.points`. Factor axes are left as-is (every level is used as a "pretty" value).

Usage:

```
R6Nomogram$pretty(w = self$x.names, v)
```

Arguments:

`w` Character vector of axis names (from `x.names`) to process. Defaults to all axes.

v Controls the chosen tick values for each axis in w. If missing, ticks are chosen automatically via axisTicks(). If a single number, it is used as the nint argument to axisTicks(). If a list, each element gives the tick values for the corresponding axis (matched by name). If a numeric vector, it is used directly as the tick values for every axis in w.

Returns: The object itself, invisibly.

R6Nomogram$pretty.y(): Choose "pretty" tick locations for the total points / linear predictor / response axis, by interpolating (via a natural spline of response on total points) the response corresponding to each chosen tick value. Results are stored in pretty.total.points, pretty.linear.predictor, and pretty.response.

Usage:

```
R6Nomogram$pretty.y(v)
```

Arguments:

v Controls the chosen total-points tick values. If missing, ticks are chosen automatically via axisTicks(). If a single number, it is used as the nint argument to axisTicks(). Otherwise, used directly as the tick values.

Returns: The object itself, invisibly.

R6Nomogram$plot(): Plot the nomogram. Draws the predictor axes (plot.x) and/or the total points / linear predictor / response axes (plot.y) using base graphics, with appearance controlled by self$options (text.par, line.par, tick.par, etc.). If predict is supplied, a prediction line is drawn connecting the predicted points on each predictor axis to the corresponding total on the response axis.

Usage:

```
R6Nomogram$plot(plot.x = TRUE, plot.y = TRUE, predict, ...)
```

Arguments:

plot.x Logical; draw the predictor axes. Default TRUE.

plot.y Logical; draw the total points / linear predictor / response axes. Default TRUE.

predict Optional named list or data frame of predictor values for which to draw a prediction line/segments on the nomogram.

... Additional graphical parameters, passed to par().

Returns: The object itself, invisibly.

R6Nomogram$grconvertX(): Convert a value from one nomogram axis scale to the points scale. For from = "Total Points", converts a total-points value to its plotting position on the points axis. For any other axis name, converts a predictor value (numeric or factor level) to its points value via spline interpolation (numeric axes) or direct lookup (factor axes).

Usage:

```
R6Nomogram$grconvertX(x, from = "Total Points", verbose)
```

Arguments:

x Value(s) to convert.

from Character; the axis to convert from. Either "Total Points" (the default) or one of x.names.

verbose Logical; print progress messages. Defaults to self$verbose.

Returns: The converted value(s), or NA (with a warning) if from does not match "Total Points" or any axis name.

`R6Nomogram$tables()`: Summarize the nomogram as a list of data frames: one data frame per predictor axis (giving `x.pretty.vals` and `x.pretty.points`), plus a final Response data frame giving `pretty.total.points`, `pretty.linear.predictor`, and `pretty.response`.

Usage:

```
R6Nomogram$tables()
```

Returns: A named list of data frames.

`R6Nomogram$clone()`: The objects of this class are cloneable with this method.

Usage:

```
R6Nomogram$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

Author(s)

Greg Snow <538280@gmail.com>

References

Regression Modeling Strategies, Frank Harrell <doi:10.1007/978-3-319-19425-7>

See Also

The nomogram function in the rms package

Examples

```
mtcars2 <- mtcars
mtcars2$cyl <- factor(mtcars2$cyl)
mtcars2$gear <- factor(mtcars2$gear)
mtcars2$vs <- factor(mtcars2$vs)

fit <- glm(mpg ~ poly(wt,2) + poly(disp,2) + cyl*gear + vs,
  data=mtcars2, family=gaussian(link=inverse))

n1 <- R6Nomogram$new(fit)
n1$plot()

n1$options$tik.len <- 0.4
n1$options$txt.pos <- 1.2
n1$options$signif.digits <- 3
n1$x.pretty.vals$`cyl:gear` <-
  c("Other", "", "", "", "6:4", "", "6:5", "8:5")
n1$pretty.y(seq(60, 200, by=10))
n1$pretty('wt', c(1.5, 2, 3, 3.5, 4, 4.25, 4.5,
                  4.75, 5, 5.25, 5.4))
```

```
n1$pretty('disp', c(75, 100, 125, 150, 175, 200, 250,
                    300, 400, 450))
n1$plot.lp <- TRUE
n1$v.pos.r <- NULL # it will be recreated properly in next plot
n1$lp.lab <- "G/M"
n1$resp.lab <- "M/G"
n1$options$text.par[['linear predictor']] <- list(cex=0.8)

n1$plot()

rm(mtcars2)
```

Index