Package: standardize (via r-universe)

October 17, 2026

Type Package

Title Tools for Standardizing Variables for Regression in R

Version 0.2.2-9999

Description Tools which allow regression variables to be placed on similar scales, offering computational benefits as well as easing interpretation of regression output.

Depends R (>= 3.6.0)

Imports lme4, MASS, methods, stats, stringr

License GPL (>= 3)

Encoding UTF-8

LazyData true

URL https://github.com/CDEager/standardize

BugReports https://github.com/CDEager/standardize/issues

RoxygenNote 7.1.0

Suggests afex, emmeans, knitr, lmerTest, rmarkdown, testthat

VignetteBuilder knitr

Config/pak/sysreqs cmake make libicu-dev

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

Date/Publication 2021-03-05 06:57:20 UTC

RemoteUrl https://github.com/cdeager/standardize

RemoteRef HEAD

RemoteSha b687af2aa09f16418ddcdb9a0b3a86118c2e4cdb

Contents

standardize-package 2
fac_and_contr 3
is.standardized 3
makepredictcall.scaledby 4

named_contr_sum 4
predict.standardized 6
print.standardized 7
ptk 8
scale_by 9
scaled_contr_poly 11
standardize 13
standardize.news 15
standardized-class 16

Index **18**

`standardize-package` *standardize: Tools for Standardizing Variables for Regression in R.*

Description

The `standardize` package provides tools for standardizing variables prior to regression (i.e. placing all of the variables to be used in a regression on similar scales). When all of the predictors in a regression are on a similar scale, it makes the interpretation of their effect sizes more comparable. In the case of gaussian regression, placing the response on unit scale also eases interpretation. Standardizing regression variables also has computational benefits in the case of mixed effects regressions, and makes determining reasonable priors in Bayesian regressions simpler. To view the package vignette, call `vignette("using-standardize", package = "standardize")`. To see the version history, call `standardize.news()`.

Details

The `named_contr_sum` function gives named sum contrasts to unordered factors, and allows the absolute value of the non-zero cells in contrast matrix to be specified through its `scale` argument. The `scaled_contr_poly` function gives orthogonal polynomial contrasts to ordered factors, and allows the standard deviation of the columns in the contrast matrix to be specified through its `scale` argument. The `scale_by` function allows numeric variables to be scaled conditioning on factors, such that the numeric variable has the same mean and standard deviation within each level of a factor (or the interaction of several factors), with the standard deviation specified through its `scale` argument.

The `standardize` function creates a `standardized` object whose elements can be used in regression fitting functions, ensuring that all of the predictors are on the same scale. This is done by passing the function's `scale` argument to `named_contr_sum` for all unordered factors (and also any predictor with only two unique values regardless of its original class), to `scaled_contr_poly` for all ordered factors, and to `scale_by` for numeric variables which contain calls to the function. For numeric predictors not contained in a `scale_by` call, `scale` is called, ensuring that the result has standard deviation equal to the `scale` argument to `standardize`. Gaussian responses are always placed on unit scale, using `scale` (or `scale_by` if the function was used on the left hand side of the regression formula). Offsets for gaussian models are divided by the standard deviation of the raw response (within-factor-level if `scale_by` is used on the response).

Author(s)

Christopher D. Eager <eager.stats@gmail.com>

`fac_and_contr` *Create a factor and specify contrasts.*

Description

The `fac_and_contr` function is a convenience function which coerces `x` to a factor with specified `levels` and `contrasts`.

Usage

```
fac_and_contr(x, levels, contrasts, ordered = FALSE)
```

Arguments

`x`	An object coercible to `factor`.
`levels`	A character vector of levels for the factor.
`contrasts`	A matrix of `contrasts` for the factor.
`ordered`	A logical indicating whether or not the factor is ordered (default `FALSE`).

Author(s)

Christopher D. Eager <eager.stats@gmail.com>

See Also

`named_contr_sum` (unordered factors) and `scaled_contr_poly` (ordered factors).

`is.standardized` *Determine if an object has class* `standardized`.

Description

Determine if an object has class `standardized`.

Usage

```
is.standardized(object)
```

Arguments

`object`	Any R object.

Value

TRUE if object is the result of a standardize call and FALSE otherwise.

Author(s)

Christopher D. Eager <eager.stats@gmail.com>

---

makepredictcall.scaledby *S3 makepredictcall method for class scaledby.*

Description

Allows scale_by to be used within a regression formula and ensures that the predvars attribute makes the correct call to scale_by.

Usage

```
## S3 method for class 'scaledby'
makepredictcall(var, call)
```

Arguments

var, call See makepredictcall.

Author(s)

Christopher D. Eager <eager.stats@gmail.com>

---

named_contr_sum *Create named sum contrasts for an unordered factor.*

Description

named_contr_sum creates sum contrasts for a factor which are named with the levels of the factor rather than with numbers (e.g. if a factor f1 has levels A, B, and C, then rather than creating contrast columns f11 and f12, it creates columns f1A and f1B). The absolute value of the non-zero elements of the matrix can also be specified.

Usage

```
named_contr_sum(x, scale = 1, return_contr = TRUE)
```

Arguments

x	An object coercible to factor or a numeric or character vector of levels.
scale	A positive number by which the entire contrast matrix returned by contr.sum is multiplied. See 'Details'.
return_contr	A logical. If TRUE (the default), a contrast matrix is returned. If FALSE, x is converted to an unordered factor with the contrast matrix applied, and the factor is returned.

Details

First, x is coerced to factor, and its levels (excluding NA) are sorted alphabetically. If there are two unique values, and they are equal to (ignoring case) "F" and "T", "FALSE" and "TRUE", "N" and "Y", "NO" and "YES", or "0" and "1", then their order is reversed (this makes it so the positive level gets the dummy coefficient rather than the negative level, yielding a more intuitive interpretation for coefficients). Then contr.sum is called, and the column names of the resulting contrast matrix are set using the character vector of unique values (excluding the final element that gets coded as -1 for all dummy variables). This entire matrix is then multiplied by scale; with the default value of 1, this does not change the matrix; if, for example, scale = 0.5, then rather than each column containing values in -1, 0, 1, each column would contain values in -0.5, 0, 0.5. If return_contr = TRUE, then this contrast matrix is returned. If return_contr = FALSE, then x is converted to an unordered factor with the named sum contrasts and returned. NA is never assigned as a level in the contrast matrix or in the factor returned by the function, but NA values in x are not removed in the factor returned when return_contr = FALSE. See the examples.

Value

If return_contr = TRUE, a contrast matrix obtained from contr.sum with named columns rather than numbered columns and deviations with magnitude scale. If return_contr = FALSE, then x is returned as an unordered factor with the named sum contrasts applied.

Author(s)

Christopher D. Eager <eager.stats@gmail.com>

See Also

scaled_contr_poly for ordered factors.

Examples

```
f <- factor(rep(c("a", "b", "c", NA), 2), levels = c("b", "c", "a"))
f <- addNA(f)
levels(f)  # NA listed as factor level
contrasts(f)  # NA included in contrast matrix
named_contr_sum(f)  # named sum contrasts (NA dropped; levels alphabetized)
named_contr_sum(levels(f))  # same output
named_contr_sum(f, return_contr = FALSE)  # factor with named sum contrasts
named_contr_sum(f, 0.5)  # deviations of magniude 0.5
```

```
f <- c(TRUE, FALSE, FALSE, TRUE)
class(f)  # logical
named_contr_sum(f)  # TRUE gets the dummy variable
f <- named_contr_sum(f, return_contr = FALSE)
class(f)  # factor

named_contr_sum(letters[1:5])  # character argument
named_contr_sum(rep(letters[1:5], 2), return_contr = FALSE)  # creates factor

# ordered factors are converted to unordered factors, so use with caution
f <- factor(rep(1:3, 2), ordered = TRUE)
is.ordered(f)  # TRUE
f
f <- named_contr_sum(f, return_contr = FALSE)
is.ordered(f)  # FALSE
f

## Not run:
# error from stats::contr.sum because only one unique non-NA value
named_contr_sum(5)
named_contr_sum(rep(c("a", NA), 3))

## End(Not run)
```

`predict.standardized` *Place new data into an already existing standardized space.*

Description

To put new data into the same standardized space as the data in the `standardized` object, `predict` can be used with the `standardized` object as the first argument. The `predict` method also allows logicals `response`, `fixed`, and `random` to be used to specify which elements of the original data frame are present in `newdata`. A regression model fit with the `formula` and `data` elements of a `standardized` object cannot be used to directly predict the response variable for new data. The new data must first be placed into the standardized space. If offsets were included in the `formula` argument used to create the `standardized` object, then when `fixed = TRUE` the offset variables must be in `newdata`. If an offset was passed to the `offset` argument in the call to `standardize`, then the offset cannot be passed to `predict`.

Usage

```
## S3 method for class 'standardized'
predict(
  object,
  newdata = NULL,
  response = FALSE,
  fixed = TRUE,
  random = TRUE,
  ...
)
```

Arguments

object	An object of class standardized.
newdata	Data to be placed into the same standardized space as the data in the call to standardize which produced the standardized object.
response	A logical (default FALSE) indicating whether newdata contains the response variable.
fixed	A logical (default TRUE) indicating whether newdata contains variables pertaining to the fixed effects.
random	A logical (default TRUE) indicating whether newdata contains variables pertaining to the random effects.
...	Ignored with a warning.

Value

A data.frame with the newdata standardized using the pred element of the standardized object.

Note

You may see a warning "contrasts dropped from factor <x>" for each factor when predicting new data with a fitted model object, but this warning can be ignored (the actual predictions will still be correct).

Author(s)

Christopher D. Eager <eager.stats@gmail.com>

Examples

```
## Not run:
train <- subset(mydata, train)
test <- subset(mydata, !train)
train.s <- standardize(y ~ x1 + f1 + (1 | g1), train)
mod <- lmer(train.s$formula, train.s$data)
test.s <- predict(train.s, test, response = TRUE)
preds <- predict(mod, newdata = test.s)  # can ignore warning about dropped contrasts
res <- test.s$y - preds

## End(Not run)
```

---

print.standardized *S3* print *method for class* standardized.

Description

S3 print method for class standardized.

Usage

```
## S3 method for class 'standardized'
print(x, ...)
```

Arguments

x	An object of class standardized.
...	Not used.

Author(s)

Christopher D. Eager <eager.stats@gmail.com>

---

ptk *Duration and voicing measures of voiceless plosives in Spanish*

Description

A dataset containing measures of total duration and voiceless period duration for instances of intervocalic Spanish /p/, /t/, and /k/. The data are taken from 18 speakers in the task dialogues in the Spanish portion of the Glissando Corpus (the speakers are university students in Valladolid, Spain). If you analyze the ptk dataset in a publication, please cite Eager (2017) from the references section below.

Usage

```
ptk
```

Format

A data frame with 751 rows and 11 variables:

cdur Total plosive duration, measured from preceding vowel intensity maximum to following vowel intensity maximum, in milliseconds.

vdur Duration of the period of voicelessness in the vowel-consonant-vowel sequence in milliseconds.

place Place of articulation (Bilabial, Dental, or Velar).

stress Syllabic stress context (Tonic, Post-Tonic, or Unstressed).

prevowel Preceding vowel phoneme identity (a, e, i, o, or u).

posvowel Following vowel phoneme identity (a, e, i, o, or u).

wordpos Position of the plosive in the word (Initial or Medial).

wordfreq Number of times the word containing the plosive occurs in the CREA corpus.

speechrate Local speech rate around the consonant in nuclei per second.

sex The speaker's sex (Female or Male).

speaker Speaker identifier (s01 through s18).

References

Eager, Christopher D. (2017). Contrast preservation and constraints on individual phonetic variation. Doctoral thesis. University of Illinois at Urbana-Champaign.

Garrido, J. M., Escudero, D., Aguilar, L., Cardeñoso, V., Rodero, E., de-la-Mota, C., ... Bonafonte, A. (2013). Glissando: a corpus for multidisciplinary prosodic studies in Spanish and Catalan. Language Resources and Evaluation, 47(4), 945–971.

Real Academia Española. Corpus de referencia del español actual (CREA). Banco de Datos. Retrieved from http://www.rae.es

De Jong, N. H., & Wempe, T. (2009). Praat script to detect syllable nuclei and measure speech rate automatically. Behavior Research Methods, 41(2), 385–390.

---

scale_by *Center and scale a continuous variable conditioning on factors.*

Description

`scale_by` centers and scales a numeric variable within each level of a factor (or the interaction of several factors).

Usage

```
scale_by(object = NULL, data = NULL, scale = 1)
```

Arguments

`object`	A `formula` whose left hand side indicates a numeric variable to be scaled and whose right hand side indicates factors to condition this scaling on; or the result of a previous call to `scale_by` or the `pred` attribute of a previous call. See 'Details'.
`data`	A data.frame containing the numeric variable to be scaled and the factors to condition on.
`scale`	Numeric (default 1). The desired standard deviation for the numeric variable within-factor-level. If the numeric variable is a matrix, then `scale` must have either one element (used for all columns), or as many elements as there are columns in the numeric variable. To center the numeric variable without scaling, set `scale` to `0`. See 'Details'.

Details

First, the behavior when `object` is a formula and `scale = 1` is described. The left hand side of the formula must indicate a numeric variable to be scaled. The full interaction of the variables on the right hand side of the formula is taken as the factor to condition scaling on (i.e. it doesn't matter whether they are separated with `+`, `:`, or `*`). For the remainder of this section, the numeric variable will be referred to as `x` and the full factor interaction term will be referred to as `facs`.

First, if facs has more than one element, then a new factor is created as their full interaction term. When a factor has NA values, NA is treated as a level. For each level of the factor which has at least two unique non-NA x values, the mean of x is recorded as the level's center and the standard deviation of x is recorded as the level's scale. The mean of these centers is recorded as new_center and the mean of these scales is recorded as new_scale, and new_center and new_scale are used as the center and scale for factor levels with fewer than two unique non-NA x values. Then for each level of the factor, the level's center is subtracted from its x values, and the result is divided by the level's scale. The result is that any level with more than two unique non-NA x values now has mean 0 and standard deviation 1, and levels with fewer than two are placed on a similar scale (though their standard deviation is undefined). Note that the overall standard deviation of the resulting variable (or standard deviations if x is a matrix) will not be exactly 1 (but will be close). The interpretation of the variable is how far an observation is from its level's average value for x in terms of within-level standard deviations.

If scale = 0, then only centering (but not scaling) is performed. If scale is neither 0 nor 1, then x is scaled such that the standard deviation within-level is scale. Note that this is different than the scale argument to scale which specifies the number the centered variable is divided by (which is the inverse of the use here). If x is a matrix with more than one column, then scale must either be a vector with an element for each column of x or a single number which will be used for all columns. If any element of scale is 0, then all elements are treated as 0. No element in scale can be negative.

If object is not a formula, it must be a numeric variable which resulted from a previous scale_by call, or the pred attribute of such a numeric variable. In this case, scale is ignored, and x in data is scaled using the formula, centers and scales in object (with new levels treated using new_center and new_scale).

Value

A numeric variable which is conditionally scaled within each level of the conditioning factor(s), with standard deviation scale. It has an additional class scaledby, as well as an attribute pred with class scaledby_pred, which is a list containing the formula, the centers and scales for known factor levels, and the center and scale to be applied to new factor levels. The variable returned can be used as the object argument in future calls to scale_by, as can its pred attribute.

Author(s)

Christopher D. Eager <eager.stats@gmail.com>

See Also

scale.

Examples

```
dat <- data.frame(
  f1 = rep(c("a", "b", "c"), c(5, 10, 20)),
  x1 = rnorm(35, rep(c(1, 2, 3), c(5, 10, 20)),
    rep(c(.5, 1.5, 3), c(5, 10, 20))))

dat$x1_scaled <- scale(dat$x1)
```

```
dat$x1_scaled_by_f1 <- scale_by(x1 ~ f1, dat)

mean(dat$x1)
sd(dat$x1)
with(dat, tapply(x1, f1, mean))
with(dat, tapply(x1, f1, sd))

mean(dat$x1_scaled)
sd(dat$x1_scaled)
with(dat, tapply(x1_scaled, f1, mean))
with(dat, tapply(x1_scaled, f1, sd))

mean(dat$x1_scaled_by_f1)
sd(dat$x1_scaled_by_f1)
with(dat, tapply(x1_scaled_by_f1, f1, mean))
with(dat, tapply(x1_scaled_by_f1, f1, sd))

newdata <- data.frame(
  f1 = c("a", "b", "c", "d"),
  x1 = rep(1, 4))

newdata$x1_pred_scaledby <- scale_by(dat$x1_scaled_by_f1, newdata)

newdata
```

`scaled_contr_poly` *Create scaled orthogonal polynomial contrasts for an ordered factor.*

Description

The function `contr.poly` creates orthogonal polynomial contrasts for an ordered factor, with the standard deviations of the columns in the contrast matrix determined by the number of columns. The `scaled_contr_poly` function takes this contrast matrix and alters the scale so that the standard deviations of the columns all equal `scale`.

Usage

```
scaled_contr_poly(x, scale = 1, return_contr = TRUE)
```

Arguments

`x`	A factor, a numeric or character vector of levels ordered least to greatest, or a single integer greater than or equal to 3. See 'Details'.
`scale`	A single positive number indicating the standard deviation for the columns of the contrast matrix. Default is 1.
`return_contr`	A logical indicating whether the contrast matrix should be returned, or `x` as an ordered factor with the contrasts applied. See 'Details'.

Details

If x is a factor, then the non-NA levels of x are used as the levels for the contrast matrix. If x is a vector, then the unique non-NA values in x in the order in which they appear in x are used as the levels for the contrast matrix. If x is a single integer greater than or equal to 3, then the numbers 1:x are used as the levels for the contrast matrix. Any other value for x results in an error (if x = 2, then polynomial contrasts are technically possible, but all binary predictors should be treated as unordered factors and coded with sum contrasts). contr.poly is then called to obtain an orthogonal polynomial contrast matrix of the appropriate degree. The contrast matrix is is put on unit scale and then multiplied by the scale argument, resulting in an orthogonal polynomial contrast matrix where each column has standard deviation scale. If return_contr = TRUE, the contrast matrix is returned. If return_contr = FALSE, then x is coerced to an ordered factor with the contrast matrix applied, and x is returned. NA is never assigned as a level in the contrast matrix or in the factor returned by the function, but NA values in x are not removed in the factor returned when return_contr = FALSE.

Value

If return_contr = TRUE a scaled orthogonal polynomial contrast matrix is returned. If return_contr = FALSE, then a factor with the scaled orthogonal polynomial contrasts is returned.

Author(s)

Christopher D. Eager <eager.stats@gmail.com>

See Also

named_contr_sum for unordered factors.

Examples

```
f <- factor(rep(c("a", "b", "c"), 5), ordered = TRUE)
contrasts(f) <- contr.poly(3)

# difference in contrasts
contrasts(f)
scaled_contr_poly(f)
scaled_contr_poly(f, scale = 0.5)

# different options for 'x'
scaled_contr_poly(levels(f))
scaled_contr_poly(3)
scaled_contr_poly(c(2, 5, 6))

# return factor
f2 <- scaled_contr_poly(f, return_contr = FALSE)
f2
```

`standardize` *Standardize a formula and data frame for regression.*

Description

Create a `standardized` object which places all variables in `data` on the same scale based on `formula`, making regression output easier to interpret. For mixed effects regressions, this also offers computational benefits, and for Bayesian regressions, it also makes determining reasonable priors easier.

Usage

```
standardize(formula, data, family = gaussian, scale = 1, offset, ...)
```

Arguments

`formula`	A regression `formula`.
`data`	A data.frame containing the variables in `formula`.
`family`	A regression `family` (default gaussian).
`scale`	The desired scale for the regression frame. Must be a single positive number. See 'Details'.
`offset`	An optional `offset` vector. Offsets can also be included in the `formula` (e.g. `y ~ x + offset(o)`), but if this is done, then the column `o` (in this example) must be in any data frame passed as the `newdata` argument to `predict`.
`...`	Currently unused. If `na.action` is specified in `...` and is anything other than `na.pass`, a warning is issued and the argument argument is ignored.

Details

First `model.frame` is called. Then, if `family = gaussian`, the response is checked to ensure that it is numeric and has more than two unique values. If `scale_by` is used on the response in `formula`, then the `scale` argument to `scale_by` is ignored and forced to `1`. If `scale_by` is not called, then `scale` is used with default arguments. The result is that gaussian responses are on unit scale (i.e. have mean `0` and standard deviation `1`), or, if `scale_by` is used on the left hand side of `formula`, unit scale within each level of the specified conditioning factor. Offsets in gaussian models are divided by the standard deviation of the the response prior to scaling (within-factor-level if `scale_by` is used on the response). In this way, if the transformed offset is added to the transformed response, and then placed back on the response's original scale, the result would be the same as if the un-transformed offset had been added to the un-transformed response. For all other values for `family`, the response and offsets are not checked. If offsets are used within the `formula`, then they will be in the `formula` and `data` elements of the `standardized` object. If the `offset` argument to the `standardize` function is used, then the offset provided in the argument will be in the `offset` element of the `standardized` object (scaled if `family = gaussian`).

For the other predictors in the formula, first any random effects grouping factors in the formula are coerced to factor and unused levels are dropped. The levels of the resulting factor are then

recorded in the groups element. Then for the remaining predictors, regardless of their original class, if they have only two unique non-NA values, they are coerced to unordered factors. Then, named_contr_sum and scaled_contr_poly are called for unordered and ordered factors, respectively, using the scale argument provided in the call to standardize as the scale argument to the contrast functions. For numeric variables, if the variable contains a call to scale_by, then, regardless of whether the call to scale_by specifies scale, the value of scale in the call to standardize is used. If the numeric variable does not contain a call to scale_by, then scale is called, ensuring that the result has standard deviation scale.

With the default value of scale = 1, the result is a standardized object which contains a formula and data frame (and offset vector if the offset argument to the standardize function was used) which can be used to fit regressions where the predictors are all on a similar scale. Its data frame has numeric variables on unit scale, unordered factors with named sum sum contrasts, and ordered factors with orthogonal polynomial contrasts on unit scale. For gaussian regressions, the response is also placed on unit scale. If scale = 0.5 (for example), then gaussian responses would still be placed on unit scale, but unordered factors' named sum contrasts would take on values -0.5, 0, 0.5 rather than -1, 0, 1, the standard deviation of each column in the contrast matrices for ordered factors would be 0.5 rather than 1, and the standard deviation of numeric variables would be 0.5 rather than 1 (within-factor-level in the case of scale_by calls).

Value

A standardized object. The formula, data, and offset elements of the object can be used in calls to regression functions.

Note

The scale_by function is supported so long as it is not nested within other function calls. The poly function is supported so long as it is either not nested within other function calls, or is nested as the transformation of the numeric variable in a scale_by call. If poly is used, then the lsmeans function will yield misleading results (as would normally be the case).

In previous versions of standardize (v0.2.0 and earlier), na.action could be specified. Starting with v0.2.1, specifying something other than na.pass is ignored with a warning. Use of na.omit and na.exclude should be done when calling regression fitting functions using the elements returned in the standardized object.

Author(s)

Christopher D. Eager <eager.stats@gmail.com>

See Also

For scaling and contrasts, see scale, scale_by, named_contr_sum, and scaled_contr_poly. For putting new data into the same space as the standardized data, see predict. For the elements in the returned object, see standardized.

Examples

```
dat <- expand.grid(ufac = letters[1:3], ofac = 1:3)
dat <- as.data.frame(lapply(dat, function(n) rep(n, 60)))
```

```
dat$ofac <- factor(dat$ofac, ordered = TRUE)
dat$x <- rpois(nrow(dat), 5)
dat$z <- rnorm(nrow(dat), rep(rnorm(30), each = 18), rep(runif(30), each = 18))
dat$subj <- rep(1:30, each = 18)
dat$y <- rnorm(nrow(dat), -2, 5)

sobj <- standardize(y ~ log(x + 1) + scale_by(z ~ subj) + ufac + ofac +
  (1 | subj), dat)

sobj
sobj$formula
head(dat)
head(sobj$data)
sobj$contrasts
sobj$groups
mean(sobj$data$y)
sd(sobj$data$y)
mean(sobj$data$log_x.p.1)
sd(sobj$data$log_x.p.1)
with(sobj$data, tapply(z_scaled_by_subj, subj, mean))
with(sobj$data, tapply(z_scaled_by_subj, subj, sd))

sobj <- standardize(y ~ log(x + 1) + scale_by(z ~ subj) + ufac + ofac +
  (1 | subj), dat, scale = 0.5)

sobj
sobj$formula
head(dat)
head(sobj$data)
sobj$contrasts
sobj$groups
mean(sobj$data$y)
sd(sobj$data$y)
mean(sobj$data$log_x.p.1)
sd(sobj$data$log_x.p.1)
with(sobj$data, tapply(z_scaled_by_subj, subj, mean))
with(sobj$data, tapply(z_scaled_by_subj, subj, sd))

## Not run:
mod <- lmer(sobj$formula, sobj$data)
# this next line causes warnings about contrasts being dropped, but
# these warnings can be ignored (i.e. the statement still evaluates to TRUE)
all.equal(predict(mod, newdata = predict(sobj, dat)), fitted(mod))

## End(Not run)
```

`standardize.news` *Print the version history of the* `standardize` *package.*

Description

Print the version history of the `standardize` package.

Usage

```
standardize.news()
```

Value

The function prints the changes and new features for each version of the package (starting with the newest version).

Author(s)

Christopher D. Eager <eager.stats@gmail.com>

Examples

```
standardize.news()
```

`standardized-class` *Class* `standardized` *containing regression variables in a standardized space.*

Description

The `standardize` function returns a list of class `standardized`, which has a `print` method, and which can additionally be used to place new data into the same standardized space as the data passed in the call to `standardize` using the `predict` function. The `standardized` list contains the following elements.

Details

call The call to `standardize` which created the object.

scale The `scale` argument to `standardize`.

formula The regression formula in standardized space (with new names) which can be used along with the `data` element to fit regressions. It has an attribute `standardized.scale` which is the same as the `scale` element of the object (this allows users and package developers to write regression-fitting functions which can tell if the input is from a `standardized` object).

family The regression family.

data A data frame containing the regression variables in a standardized space (renamed to have valid variable names corresponding to those in the `formula` element).

offset The offset passed through the `offset` argument to `standardize` (scaled if `family = gaussian`), or `NULL` if the `offset` argument was not used.

pred A list containing unevaluated calls which allow the `predict` method to work.

variables A data frame with the name of the original variable, the corresponding name in the standardized data frame and formula, and the class of the variable in the standardized data frame.

contrasts A named list of contrasts for all factors included as predictors, or NULL if no predictors are factors.

groups A named list of levels for random effects grouping factors, or NULL if there are no random effects.

In the variables data frame, the Variable column contains the name of the variable in the original formula passed to standardize. The Standardized Name column contains the name of the variable in the standardized formula and data frame. The original variable name is altered such that the original name is still recoverable but is also a valid variable name for regressions run using the formula and data elements of the standardized object. For example, exp(x) would become exp_x and log(x + 1) would become log_x.p.1. If the indicator function is used, this can lead to a long and possibly difficult to interpret name; e.g. I(x1 > 0 & x2 < 0) would become I_x1.g.0.a.x2.l.0. In such cases, it is better to create the variable explicitly in the data frame and give it a meaningful name; in this case, something like mydata$x1Pos_x2Neg <- mydata$x1 > 0 & mydata$x2 < 0, and then use x1Pos_x2Neg in the call to standardize. The Class column in the variables data frame takes the following values (except for non-gaussian responses, which are left unaltered, and so may have a different class; the class for the response is always preceded by response.).

numeric A numeric vector.

poly A numeric matrix resulting from a call to poly.

scaledby A numeric vector resulting from a call to scale_by.

scaledby.poly A numeric matrix resulting from a call to poly nested within a call to scale_by.

factor An unordered factor.

ordered An ordered factor.

group A random effects grouping factor.

offset If the offset function was used within the formula passed to standardize, then the variable is numeric and labeled as offset. The formula element of the standardize object contains offset calls to ensure regression fitting functions use them properly. If the offset argument was used in the call to standardize (rather than putting offset calls in the formula), then the offset is not in the variables data frame (it is in the offset element of the standardized object).

The standardized object has a printing method which displays the call, formula, and variable frame along with an explanation of the standardization. The is.standardized function returns TRUE if an object is the result of a call to standardize and FALSE otherwise. The predict method places new data into the same standardized space as the data passed to the original standardize call.

Author(s)

Christopher D. Eager <eager.stats@gmail.com>

Index

∗ **datasets**
ptk, 8

contr.poly, *11, 12*
contr.sum, *5*
contrasts, *3*

fac_and_contr, 3
factor, *3*
family, *13*
formula, *4*, *9*, *13*

is.standardized, 3, *17*

makepredictcall, *4*
makepredictcall.scaledby, 4
model.frame, *13*

named_contr_sum, *2, 3*, 4, *12*, *14*

offset, *13*

poly, *14*, *17*
predict, *13, 14*, *16, 17*
predict.standardized, 6
print.standardized, 7
ptk, 8

scale, *2*, *10*, *13, 14*
scale_by, *2*, *4*, 9, *13, 14*, *17*
scaled_contr_poly, *2, 3*, *5*, 11, *14*
standardize, *2*, *4*, *6, 7*, 13, *16, 17*
standardize-package, 2
standardize.news, 15
standardized, *2, 3*, *6, 7*, *13, 14*
standardized-class, 16