Package ‘polypoly’

May 27, 2017

Title Helper Functions for Orthogonal Polynomials
Version 0.0.2
Description Tools for reshaping, plotting, and manipulating matrices of orthogonal polynomials.
Depends R (>= 3.3.3)
License GPL-3
Encoding UTF-8
LazyData true
URL https://github.com/tjmahr/polypoly
BugReports https://github.com/tjmahr/polypoly/issues
Imports tibble, reshape2, ggplot2, rlang, stats
RoxygenNote 6.0.1
Suggests testthat, knitr, rmarkdown, lme4, splines
VignetteBuilder knitr
NeedsCompilation no
Author Tristan Mahr [aut, cre]
Maintainer Tristan Mahr <tristan.mahr@gmail.com>
Repository CRAN
Date/Publication 2017-05-27 14:12:20 UTC

R topics documented:

polypoly .......................................................... 2
poly_add_columns ................................................ 2
poly_melt .......................................................... 3
poly_plot .......................................................... 3
poly_rescale ....................................................... 4

Index 6
poly_poly

polypoly: Helper functions for orthogonal polynomials

Description

This package provides helpful functions for orthogonal polynomials created by stats::poly(). These include plotting poly_plot(), tidying poly_melt(), rescaling poly_rescale(), and manipulating a dataframe poly_add_columns().

Author(s)

Tristan Mahr

poly_add_columns

Add orthogonal polynomial columns to a dataframe

Description

Add orthogonal polynomial columns to a dataframe

Usage

poly_add_columns(.data, .col, degree = 1L, prefix = nullL, scale_width = nullL)

Arguments

.data

a dataframe

.col

a bare column name

degree

number of polynomial terms to add to the dataframe

prefix

prefix for the names to add to the dataframe. default is the name of .col.

scale_width

optionally rescale the dataframe using poly_rescale(). Default behavior is not to perform any rescaling.

Value

the dataframe with additional columns of orthogonal polynomial terms of .col

Examples

df <- data.frame(time = rep(1:5, 3), y = rnorm(15))

# adds columns "time1", "time2", "time3"
poly_add_columns(df, time, degree = 3)

# adds columns "t1", "t2", "t3" and rescale
poly_add_columns(df, time, degree = 3, prefix = "t", scale_width = 1)
**poly_melt**

*Melt a polynomial matrix*

**Description**

Melt a polynomial matrix

**Usage**

```r
poly_melt(x)
```

**Arguments**

- `x` a matrix created by `stats::poly()`

**Details**

The degree values are returned as a character vector because they should be treated categorically (as when plotting). Moreover, matrices made with multiple vectors (e.g., `poly(rnorm(10), rnorm(10), degree = 2)`) have names that are not numerically meaningful (e.g., `1.0, 2.0, 0.1, 1.1, 0.2`).

**Value**

A `tibble::tibble()` with three columns: observation (row number of the matrix), polynomial degree, and value.

**Examples**

```r
m <- poly(rnorm(10), degree = 3)
poly_melt(m)
```

---

**poly_plot**

*Plot a polynomial matrix*

**Description**

Plot a polynomial matrix

**Usage**

```r
poly_plot(x, by_observation = TRUE, x_col = 1)
poly_plot_data(x, by_observation = TRUE, x_col = 1)
```
poly_rescale

Description
Rescale the range of a polynomial matrix

Usage
poly_rescale(x, scale_width = 1)

Arguments
- x: a matrix created by stats::poly()
- scale_width: the desired range (max - min) for the first column of the matrix

Details
This function strips away the poly class and the coef attribute of the matrix. This is because those attributes no longer describe the transformed matrix.
**poly_rescale**

**Value**

the rescaled polynomial matrix (as a plain matrix with coefs attribute removed)

**Examples**

```r
m <- poly(1:10, degree = 4)

# Difference between min and max values of first column is 10
scaled <- poly_rescale(m, scale_width = 10)
scaled

# Rescaled values are still orthogonal
zapsmall(cor(scaled))
```
Index

ggplot2::ggplot(), 4
poly_add_columns, 2
poly_add_columns(), 2
poly_melt, 3
poly_melt(), 2
poly_plot, 3
poly_plot(), 2
poly_plot_data (poly_plot), 3
poly_rescale, 4
poly_rescale(), 2
polypoly, 2
polypoly-package (polypoly), 2
stats::poly(), 2–4
tibble::tibble(), 3