

# Package: polypoly (via r-universe)

September 11, 2024

**Title** Helper Functions for Orthogonal Polynomials

**Version** 0.0.3

**Description** Tools for reshaping, plotting, and manipulating matrices of orthogonal polynomials.

**Depends** R (>= 3.3.3)

**License** GPL-3

**Encoding** UTF-8

**Roxygen** list(markdown = TRUE)

**URL** <https://github.com/tjmahr/polypoly>

**BugReports** <https://github.com/tjmahr/polypoly/issues>

**Imports** tibble, reshape2, ggplot2, rlang, stats

**RoxygenNote** 7.2.1

**Suggests** testthat, knitr, rmarkdown, lme4, splines

**VignetteBuilder** knitr

**Repository** <https://tjmahr.r-universe.dev>

**RemoteUrl** <https://github.com/tjmahr/polypoly>

**RemoteRef** HEAD

**RemoteSha** 5eae95f1efa40ebadc31c33918521d462a9dbf45

## Contents

polypoly . . . . .	2
poly_add_columns . . . . .	2
poly_melt . . . . .	3
poly_plot . . . . .	4
poly_rescale . . . . .	5

<b>Index</b>	<b>6</b>
--------------	----------

---

 polypoly

*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 = 1,
  prefix = NULL,
  scale_width = NULL,
  na_values = c("error", "warn", "allow")
)
```

### Arguments

<code>.data</code>	a dataframe
<code>.col</code>	a bare column name
<code>degree</code>	number of polynomial terms to add to the dataframe
<code>prefix</code>	prefix for the names to add to the dataframe. default is the name of <code>.col</code> .
<code>scale_width</code>	optionally rescale the dataframe using <code>poly_rescale()</code> . Default behavior is not to perform any rescaling.
<code>na_values</code>	How to handle missing values. Default is "error" which raises an error. Other options include "warn" to raise a warning and "allow" to silently accept missing values.

**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**

```
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**

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

---

poly\_plot

*Plot a polynomial matrix*

---

### Description

Plot a polynomial matrix

### Usage

```
poly_plot(x, by_observation = TRUE, x_col = 1)
```

```
poly_plot_data(x, by_observation = TRUE, x_col = 1)
```

### Arguments

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

**by\_observation** whether the x axis should be mapped to the observation/row number (TRUE, the default) or to the degree-1 terms of the matrix (FALSE)

**x\_col** integer indicating which column to plot as the x-axis when `by_observation` is FALSE. Default is 1 (assumes the first column is the linear polynomial term).

### Value

a `ggplot2::ggplot()` plot of the degree terms from the matrix. For `poly_plot_data()`, the dataframe used to create the plot is returned instead.

### Examples

```
# Defaults to plotting using the row number as x-axis
m <- poly(1:100, degree = 3)
poly_plot(m)

# Not good because observations were not sorted
m2 <- poly(rnorm(100), degree = 3)
poly_plot(m2)

# Instead set by_observation to FALSE to plot along the degree 1 values
poly_plot(m2, by_observation = FALSE)

# Get a dataframe instead of plot
poly_plot_data(m2, by_observation = FALSE)
```

---

poly_rescale	<i>Rescale the range of a polynomial matrix</i>
--------------	---

---

**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 `coefs` attribute of the matrix. This is because those attributes no longer describe the transformed matrix.

**Value**

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

**Examples**

```
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`, 4

`poly_plot()`, 2

`poly_plot_data (poly_plot)`, 4

`poly_rescale`, 5

`poly_rescale()`, 2

`polypoly`, 2

`stats::poly()`, 2–5

`tibble::tibble()`, 3