Package: standard (via r-universe)

September 18, 2024

Title Simplified Fitting and Use of Standard Curves

Version 0.1.0

Description {standard} provices a more simplified interface to the linear models system in R for the fitting of standard curves and their usage in biochemistry and molecular biology.

License MIT + file LICENSE

Suggests covr, roxygen2, testthat (>= 3.0.0), vdiffr

Config/testthat/edition 3

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.1

Imports tibble, broom, purrr, rlang (>= 0.4.11), dplyr, ggpp, stringr, ggtext, ggplot2, methods

URL https://github.com/bradyajohnston/standard

BugReports https://github.com/bradyajohnston/standard/issues

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

RemoteUrl https://github.com/bradyajohnston/standard

RemoteRef HEAD

RemoteSha 20602c303650cd2835b0d860e837b094a7fcab29

Contents

`as.data.frame.std_calc` *Convert* `std_calc` *to data frame*

Description

Convert `std_calc` to data frame

Usage

```
## S3 method for class 'std_calc'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)
```

Arguments

`x`	object of class `std_calc`, the output of `std_curve_calc()`
`row.names`	Optional vector of rownames.
`optional`	logical. If TRUE, setting row names and converting column names (to syntactic names: see make.names) is optional. Note that all of R's base package as.data.frame() methods use optional only for column names treatment, basically with the meaning of data.frame(*, check.names = !optional). See also the make.names argument of the matrix method.
`...`	additional arguments to be passed to or from methods.

Value

data.frame

lerp *Title*

Description

Title

Usage

```
lerp(x, y, z = 0.5)
```

Arguments

x	Value to lerp from.
y	Value to lerp to.
z	Proportion to lerp by (0-1).

lerp_vec *Lerp the Max and Min of a Vector.*

Description

Finds the minium value of a vector and the maximum value of a vector, and then lerps between the two by the factor z.

Usage

```
lerp_vec(vec, z = 0.5)
```

Arguments

vec	numeric vector of values.
z	proportion to lerp by (0-1).

n_decimal *Finds Minimum Number of Decimal Places*

Description

Finds Minimum Number of Decimal Places

Usage

```
n_decimal(x)
```

Arguments

x	Number to calculate decimal places.

plot.std_calc *Generic Function for Plotting Standard Curve Calculations*

Description

Generic Function for Plotting Standard Curve Calculations

Usage

```
## S3 method for class 'std_calc'
plot(x, ...)
```

Arguments

x	output of `std_curve_calc()`
...	Additional arguments to be passed to `std_curve_plot()`

Value

ggplot2 plot

plot.std_curve *Generic Function for Plotting Fitted Standard Curves*

Description

Generic Function for Plotting Fitted Standard Curves

Usage

```
## S3 method for class 'std_curve'
plot(x, ...)
```

Arguments

x	output of std_surve_fit()
...	Additional arguments to be passed to std_curve_plot()

Value

ggplot2 plot

print.std_calc *Printing Results of* std_curve_calc()

Description

Printing Results of std_curve_calc()

Usage

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

Arguments

x	object of class std_calc, the output of std_curve_calc
...	additional arguments to be passed to or from methods.

`std_curve_calc` *Use a Standard Curve to Calculate Unknown Values*

Description

Use a Standard Curve to Calculate Unknown Values

Usage

```
std_curve_calc(std_curve, unknowns, digits = 3)
```

Arguments

`std_curve`	A linear model, created with either `lm()` or `standard::std_curve_fit()`
`unknowns`	A numeric vector of unknown values, which the standard curve will be used to predict their values.
`digits`	Number of decimal places for calculations.

Value

a tibble with a column for the unknown values, and a column `.fitted` for the predicted values, based on the standard curve.

Examples

```
library(standard)

# Protein concentrations of the standards used in the assay
prot <- c(
  0.000, 0.016, 0.031, 0.063, 0.125, 0.250, 0.500, 1.000,
  0.000, 0.016, 0.031, 0.063, 0.125, 0.250, 0.500, 1.000
)

# absorbance readins from the standards used in the assay
abs <- c(
  0.329, 0.352, 0.349, 0.379, 0.417, 0.491, 0.668, 0.956,
  0.327, 0.341, 0.355, 0.383, 0.417, 0.446, 0.655, 0.905
)
assay_data <- data.frame(
  Protein = prot,
  Absorbance = abs
)

# unknown concentrations
unk <- c(0.554, 0.568, 0.705)


assay_data |>
  std_curve_fit(Protein, Absorbance) |>
```

```
std_curve_calc(unk) |>
plot()
```

`std_curve_fit` *Create a Standard Curve From Known Data*

Description

Create a Standard Curve From Known Data

Usage

```
std_curve_fit(data, conc, resp)
```

Arguments

`data`	A `data.frame` that contains the columns for concentration and observed response for the standard curve.
`conc`	Name of the column that contains the concentration for the standard curve.
`resp`	Name of the column that contains the response values for the standard curve.

Value

A linear model (lm()) object to be used as a standard curve, for use with `standard::std_curve_calc()` `broom::augment()` or `stats::predict()`.

Examples

```
library(standard)

# Protein concentrations of the standards used in the assay
prot <- c(
  0.000, 0.016, 0.031, 0.063, 0.125, 0.250, 0.500, 1.000,
  0.000, 0.016, 0.031, 0.063, 0.125, 0.250, 0.500, 1.000
)

# absorbance readings from the standards used in the assay
abs <- c(
  0.329, 0.352, 0.349, 0.379, 0.417, 0.491, 0.668, 0.956,
  0.327, 0.341, 0.355, 0.383, 0.417, 0.446, 0.655, 0.905
)
assay_data <- data.frame(
  Protein = prot,
  Absorbance = abs
)

# unknown concentrations
unk <- c(0.554, 0.568, 0.705)
```

```
assay_data |>
  std_curve_fit(Protein, Absorbance) |>
  plot()
```

`std_curve_plot` *Plot a Standard Curve*

Description

Plot a Standard Curve

Usage

```
std_curve_plot(data)
```

Arguments

`data`	Result of `std_curve_pred()` or `std_curve_fit()`.

Value

a ggplot2 plot with the standard curve and unkowns plotted, whch can be further customised using `ggplot` options.

Examples

```
library(standard)

# Protein concentrations of the standards used in the assay
prot <- c(
  0.000, 0.016, 0.031, 0.063, 0.125, 0.250, 0.500, 1.000,
  0.000, 0.016, 0.031, 0.063, 0.125, 0.250, 0.500, 1.000
)

# absorbance readins from the standards used in the assay
abs <- c(
  0.329, 0.352, 0.349, 0.379, 0.417, 0.491, 0.668, 0.956,
  0.327, 0.341, 0.355, 0.383, 0.417, 0.446, 0.655, 0.905
)
assay_data <- data.frame(
  Protein = prot,
  Absorbance = abs
)

# unknown concentrations
unk <- c(0.554, 0.568, 0.705)


assay_data |>
```

```
  std_curve_fit(Protein, Absorbance) |>
  std_curve_calc(unk) |>
  plot()
```

`std_paste_formula` *Extract and Paste Formula From Standard Curve*

Description

Extract and Paste Formula From Standard Curve

Usage

```
std_paste_formula(std_curve, digits = 3)
```

Arguments

`std_curve`	object of class std_curve, the output of `std_curve_fit()`
`digits`	Number of decimal places to round numbers in the formula to.

Value

a string of the extracted formula from the standard curve

Examples

```
library(standard)

# Protein concentrations of the standards used in the assay
prot <- c(
  0.000, 0.016, 0.031, 0.063, 0.125, 0.250, 0.500, 1.000,
  0.000, 0.016, 0.031, 0.063, 0.125, 0.250, 0.500, 1.000
)

# absorbance readins from the standards used in the assay
abs <- c(
  0.329, 0.352, 0.349, 0.379, 0.417, 0.491, 0.668, 0.956,
  0.327, 0.341, 0.355, 0.383, 0.417, 0.446, 0.655, 0.905
)
assay_data <- data.frame(
  Protein = prot,
  Absorbance = abs
)

# unknown concentrations
unk <- c(0.554, 0.568, 0.705)


assay_data |>
  std_curve_fit(Protein, Absorbance) |>
  std_paste_formula()
```

`[.std_calc` *Generic function for subsetting output of* `std_curve_fit()`

Description

Generic function for subsetting output of `std_curve_fit()`

Usage

```
## S3 method for class 'std_calc'
x[i, j]
```

Arguments

`x`	object of class `std_curve`, the output of `std_curve_fit()`
`i`	row index
`j`	column index

Value

column of tibble

Index