Package 'scales'

July 2, 2014

Type Package

Title Scale functions for graphics.

Version 0.2.4

Author Hadley Wickham <h.wickham@gmail.com>

Maintainer Hadley Wickham <h.wickham@gmail.com>

Description Scales map data to aesthetics, and provide
methods for automatically determining breaks and labels for axes and legends.

URL https://github.com/hadley/scales

BugReports https://github.com/hadley/scales/issues

Depends R (>= 2.13)

Imports RColorBrewer, dichromat, munsell (>= 0.2), plyr (>= 1.2),labeling, methods

Suggests testthat (>= 0.8)

License MIT + file LICENSE

LazyLoad yes

BuildManual yes

NeedsCompilation no

Repository CRAN

Date/Publication 2014-04-22 10:50:55

R topics documented:

`abs_area` *Point area palette (continuous), with area proportional to value.*

Description

Point area palette (continuous), with area proportional to value.

Usage

```
abs_area(max)
```

Arguments

`max`	A number representing the maxmimum size.

`alpha` *Modify colour transparency. Vectorised in both colour and alpha.*

Description

Modify colour transparency. Vectorised in both colour and alpha.

Usage

```
alpha(colour, alpha = NA)
```

Arguments

colour	colour
alpha	new alpha level in [0,1]. If alpha is NA, existing alpha values are preserved.

Examples

```
alpha("red", 0.1)
alpha(colours(), 0.5)
alpha("red", seq(0, 1, length = 10))
```

---

area_pal *Point area palette (continuous).*

Description

Point area palette (continuous).

Usage

```
area_pal(range = c(1, 6))
```

Arguments

range	Numeric vector of length two, giving range of possible sizes. Should be greater than 0.

---

as.trans *Convert character string to transformer.*

Description

Convert character string to transformer.

Usage

```
as.trans(x)
```

Arguments

x	name of transformer

asn_trans *Arc-sin square root transformation.*

Description

Arc-sin square root transformation.

Usage

```
asn_trans()
```

atanh_trans *Arc-tangent transformation.*

Description

Arc-tangent transformation.

Usage

```
atanh_trans()
```

boxcox_trans *Box-Cox power transformation.*

Description

Box-Cox power transformation.

Usage

```
boxcox_trans(p)
```

Arguments

p	Exponent of boxcox transformation.

References

See http://en.wikipedia.org/wiki/Power_transform for

brewer_pal *Color Brewer palette (discrete).*

Description

Color Brewer palette (discrete).

Usage

```
brewer_pal(type = "seq", palette = 1)
```

Arguments

type	One of seq (sequential), div (diverging) or qual (qualitative)
palette	If a string, will use that named palette. If a number, will index into the list of palettes of appropriate type

References

http://colorbrewer2.org

Examples

```
show_col(brewer_pal()(10))
show_col(brewer_pal("div")(5))
show_col(brewer_pal(pal = "Greens")(5))

# Can use with gradient_n to create a continous gradient
cols <- brewer_pal("div")(5)
show_col(gradient_n_pal(cols)(seq(0, 1, length = 30)))
```

cbreaks *Compute breaks for continuous scale.*

Description

This function wraps up the components needed to go from a continuous range to a set of breaks and labels suitable for display on axes or legends.

Usage

```
cbreaks(range, breaks = extended_breaks(), labels = scientific_format())
```

Arguments

range	numeric vector of length 2 giving the range of the underlying data
breaks	either a vector of break values, or a break function that will make a vector of breaks when given the range of the data
labels	either a vector of labels (character vector or list of expression) or a format function that will make a vector of labels when called with a vector of breaks. Labels can only be specified manually if breaks are - it is extremely dangerous to supply labels if you don't know what the breaks will be.

Examples

```
cbreaks(c(0, 100))
cbreaks(c(0, 100), pretty_breaks(3))
cbreaks(c(0, 100), pretty_breaks(10))
cbreaks(c(1, 100), log_breaks())
cbreaks(c(1, 1e4), log_breaks())

cbreaks(c(0, 100), labels = math_format())
cbreaks(c(0, 1), labels = percent_format())
cbreaks(c(0, 1e6), labels = comma_format())
cbreaks(c(0, 1e6), labels = dollar_format())
cbreaks(c(0, 30), labels = dollar_format())

# You can also specify them manually:
cbreaks(c(0, 100), breaks = c(15, 20, 80))
cbreaks(c(0, 100), breaks = c(15, 20, 80), labels = c(1.5, 2.0, 8.0))
cbreaks(c(0, 100), breaks = c(15, 20, 80),
  labels = expression(alpha, beta, gamma))
```

---

censor *Censor any values outside of range.*

Description

Censor any values outside of range.

Usage

```
censor(x, range = c(0, 1), only.finite = TRUE)
```

Arguments

x	numeric vector of values to manipulate.
range	numeric vector of length two giving desired output range.
only.finite	if TRUE (the default), will only modify finite values.

Examples

```
censor(c(-1, 0.5, 1, 2, NA))
```

col2hcl *Modify standard R colour in hcl colour space.*

Description

Transforms rgb to hcl, sets non-missing arguments and then backtransforms to rgb.

Usage

```
col2hcl(colour, h, c, l, alpha = 1)
```

Arguments

colour	character vector of colours to be modified
h	new hue
l	new luminance
c	new chroma
alpha	alpha value. Defaults to 1.

Examples

```
col2hcl(colors())
```

comma_format *Comma formatter: format number with commas separating thousands.*

Description

Comma formatter: format number with commas separating thousands.

Usage

```
comma_format(...)

comma(x, ...)
```

Arguments

...	other arguments passed on to format
x	a numeric vector to format

Value

a function with single paramater x, a numeric vector, that returns a character vector

Examples

```
comma_format()(c(1, 1e3, 2000, 1e6))
comma_format(digits = 9)(c(1, 1e3, 2000, 1e6))
comma(c(1, 1e3, 2000, 1e6))
```

---

cscale *Continuous scale.*

Description

Continuous scale.

Usage

```
cscale(x, palette, na.value = NA_real_, trans = identity_trans())
```

Arguments

x	vector of continuous values to scale
palette	palette to use. Built in palettes: area_pal, brewer_pal, dichromat_pal, div_gradient_pal, gradient_n_pal, grey_pal, hue_pal, identity_pal, linetype_pal, manual_pal, rescale_pal, seq_gradient_pal, shape_pal
na.value	value to use for missing values
trans	transformation object describing the how to transform the raw data prior to scaling. Defaults to the identity transformation which leaves the data unchanged. Built in transformations: asn_trans, atanh_trans, boxcox_trans, date_trans, exp_trans, identity_trans, log10_trans, log1p_trans, log2_trans, log_trans, logit_trans, probability_trans, probit_trans, reciprocal_trans, reverse_trans, sqrt_trans, time_trans.

Examples

```
with(mtcars, plot(disp, mpg, cex = cscale(hp, rescale_pal())))
with(mtcars, plot(disp, mpg, cex = cscale(hp, rescale_pal(),
  trans = sqrt_trans())))
with(mtcars, plot(disp, mpg, cex = cscale(hp, area_pal())))
with(mtcars, plot(disp, mpg, pch = 20, cex = 5,
  col = cscale(hp, seq_gradient_pal("grey80", "black"))))
```

date_breaks *Regularly spaced dates.*

Description

Regularly spaced dates.

Usage

```
date_breaks(width = "1 month")
```

Arguments

width	an interval specification, one of "sec", "min", "hour", "day", "week", "month", "year". Can be by an integer and a space, or followed by "s".

date_format *Formatted dates.*

Description

Formatted dates.

Usage

```
date_format(format = "%Y-%m-%d")
```

Arguments

format	Date format using standard POSIX specification. See strptime for possible formats.

date_trans *Transformation for dates (class Date).*

Description

Transformation for dates (class Date).

Usage

```
date_trans()
```

Examples

```
years <- seq(as.Date("1910/1/1"), as.Date("1999/1/1"), "years")
t <- date_trans()
t$trans(years)
t$inv(t$trans(years))
t$format(t$breaks(range(years)))
```

`dichromat_pal` *Dichromat (colour-blind) palette (discrete).*

Description

Dichromat (colour-blind) palette (discrete).

Usage

```
dichromat_pal(name)
```

Arguments

`name`	Name of colour palette. One of: `BrowntoBlue.10`, `BrowntoBlue.12`, `BluetoDarkOrange.12`, `BluetoDarkOrange.18`, `DarkRedtoBlue.12`, `DarkRedtoBlue.18`, `BluetoGreen.14`, `BluetoGray.8`, `BluetoOrangeRed.14`, `BluetoOrange.10`, `BluetoOrange.12`, `BluetoOrange.8`, `LightBluetoDarkBlue.10`, `LightBluetoDarkBlue.7`, `Categorical.12`, `GreentoMagenta.16`, `SteppedSequential.5`

Examples

```
show_col(dichromat_pal("BluetoOrange.10")(10))
show_col(dichromat_pal("BluetoOrange.10")(5))

# Can use with gradient_n to create a continous gradient
cols <- dichromat_pal("DarkRedtoBlue.12")(12)
show_col(gradient_n_pal(cols)(seq(0, 1, length = 30)))
```

`discard` *Discard any values outside of range.*

Description

Discard any values outside of range.

Usage

```
discard(x, range = c(0, 1))
```

Arguments

x	numeric vector of values to manipulate.
range	numeric vector of length two giving desired output range.

Examples

```
discard(c(-1, 0.5, 1, 2, NA))
```

---

div_gradient_pal *Diverging colour gradient (continous).*

Description

Diverging colour gradient (continous).

Usage

```
div_gradient_pal(low = mnsl("10B 4/6"), mid = mnsl("N 8/0"),
  high = mnsl("10R 4/6"), space = "Lab")
```

Arguments

low	colour for low end of gradient.
mid	colour for mid point
high	colour for high end of gradient.
space	colour space in which to calculate gradient. "Lab" usually best unless gradient goes through white.

Examples

```
x <- seq(-1, 1, length = 100)
r <- sqrt(outer(x^2, x^2, "+"))
image(r, col = div_gradient_pal()(seq(0, 1, length = 12)))
image(r, col = div_gradient_pal()(seq(0, 1, length = 30)))
image(r, col = div_gradient_pal()(seq(0, 1, length = 100)))

library(munsell)
image(r, col = div_gradient_pal(low =
   mnsl(complement("10R 4/6", fix = TRUE)))(seq(0, 1, length = 100)))
```

dollar_format *Currency formatter: round to nearest cent and display dollar sign.*

Description

The returned function will format a vector of values as currency. Values are rounded to the nearest cent, and cents are displayed if any of the values has a non-zero cents and the largest value is less than `largest_with_cents` which by default is 100000.

Usage

```
dollar_format(largest_with_cents = 1e+05)

dollar(x)
```

Arguments

`largest_with_cents`	the value that all values of `x` must be less than in order for the cents to be displayed
`x`	a numeric vector to format

Value

a function with single paramater x, a numeric vector, that returns a character vector

Examples

```
dollar_format()(c(100, 0.23, 1.456565, 2e3))
dollar_format()(c(1:10 * 10))
dollar(c(100, 0.23, 1.456565, 2e3))
dollar(c(1:10 * 10))
dollar(10^(1:8))
```

dscale *Discrete scale.*

Description

Discrete scale.

Usage

```
dscale(x, palette, na.value = NA)
```

Arguments

x	vector of discrete values to scale
palette	aesthetic palette to use
na.value	aesthetic to use for missing values

Examples

```
with(mtcars, plot(disp, mpg, pch = 20, cex = 3,
  col = dscale(factor(cyl), brewer_pal())))
```

---

expand_range *Expand a range with a multiplicative or additive constant.*

Description

Expand a range with a multiplicative or additive constant.

Usage

```
expand_range(range, mul = 0, add = 0, zero_width = 1)
```

Arguments

range	range of data, numeric vector of length 2
mul	multiplicative constract
add	additive constant
zero_width	distance to use if range has zero width

---

exp_trans *Exponential transformation (inverse of log transformation).*

Description

Exponential transformation (inverse of log transformation).

Usage

```
exp_trans(base = exp(1))
```

Arguments

base	Base of logarithm

extended_breaks *Extended breaks. Uses Wilkinson's extended breaks algorithm as implemented in the* **labeling** *package.*

Description

Extended breaks. Uses Wilkinson's extended breaks algorithm as implemented in the **labeling** package.

Usage

```
extended_breaks(n = 5, ...)
```

Arguments

n	desired number of breaks
...	other arguments passed on to extended

References

Talbot, J., Lin, S., Hanrahan, P. (2010) An Extension of Wilkinson's Algorithm for Positioning Tick Labels on Axes, InfoVis 2010.

Examples

```
extended_breaks()(1:10)
extended_breaks()(1:100)
```

format_format *Format with using any arguments to* format.

Description

If the breaks have names, they will be used in preference to formatting the breaks.

Usage

```
format_format(...)
```

Arguments

...	other arguments passed on to format.

See Also

format, format.Date, format.POSIXct

gradient_n_pal *Arbitrary colour gradient palette (continous).*

Description

Arbitrary colour gradient palette (continous).

Usage

```
gradient_n_pal(colours, values = NULL, space = "Lab")
```

Arguments

colours	vector of colours
values	if colours should not be evenly positioned along the gradient this vector gives the position (between 0 and 1) for each colour in the colours vector. See rescale for a convience function to map an arbitrary range to between 0 and 1.
space	colour space in which to calculate gradient. "Lab" usually best unless gradient goes through white.

grey_pal *Grey scale palette (discrete).*

Description

Grey scale palette (discrete).

Usage

```
grey_pal(start = 0.2, end = 0.8)
```

Arguments

start	gray value at low end of palette
end	gray value at high end of palette

See Also

seq_gradient_pal for continuous version

Examples

```
show_col(grey_pal()(25))
show_col(grey_pal(0, 1)(25))
```

hue_pal *Hue palette (discrete).*

Description

Hue palette (discrete).

Usage

```
hue_pal(h = c(0, 360) + 15, c = 100, l = 65, h.start = 0,
  direction = 1)
```

Arguments

h	range of hues to use, in [0, 360]
l	luminance (lightness), in [0, 100]
c	chroma (intensity of colour), maximum value varies depending on
h.start	hue to start at
direction	direction to travel around the colour wheel, 1 = clockwise, -1 = counter-clockwise

Examples

```
show_col(hue_pal()(4))
show_col(hue_pal()(9))
show_col(hue_pal(l = 90)(9))
show_col(hue_pal(l = 30)(9))

show_col(hue_pal()(9))
show_col(hue_pal(direction = -1)(9))

show_col(hue_pal()(9))
show_col(hue_pal(h = c(0, 90))(9))
show_col(hue_pal(h = c(90, 180))(9))
show_col(hue_pal(h = c(180, 270))(9))
show_col(hue_pal(h = c(270, 360))(9))
```

identity_pal *Identity palette.*

Description

Leaves values unchanged - useful when the data is already scaled.

Usage

```
identity_pal()
```

identity_trans *Identity transformation (do nothing).*

Description

Identity transformation (do nothing).

Usage

```
identity_trans()
```

linetype_pal *Line type palette (discrete).*

Description

Based on a set supplied by Richard Pearson, University of Manchester

Usage

```
linetype_pal()
```

log1p_trans *Log plus one transformation.*

Description

Log plus one transformation.

Usage

```
log1p_trans()
```

log_breaks *Log breaks (integer breaks on log-transformed scales).*

Description

Log breaks (integer breaks on log-transformed scales).

Usage

```
log_breaks(n = 5, base = 10)
```

Arguments

n	desired number of breaks
base	base of logarithm to use

Examples

```
log_breaks()(c(1, 1e6))
log_breaks()(c(1, 1e5))
```

log_trans *Log transformation.*

Description

Log transformation.

Usage

```
log_trans(base = exp(1))
```

Arguments

base	base of logarithm

`manual_pal` *Manual palette (manual).*

Description

Manual palette (manual).

Usage

```
manual_pal(values)
```

Arguments

`values`	vector of values to be used as a palette.

`math_format` *Add arbitrary expression to a label. The symbol that will be replace by the label value is* `.x`.

Description

Add arbitrary expression to a label. The symbol that will be replace by the label value is `.x`.

Usage

```
math_format(expr = 10^.x, format = force)
```

Arguments

`expr`	expression to use
`format`	another format function to apply prior to mathematical transformation - this makes it easier to use floating point numbers in mathematical expressions.

Value

a function with single paramater x, a numeric vector, that returns a list of expressions

See Also

`plotmath`

Examples

```
math_format()(1:10)
math_format(alpha + frac(1, .x))(1:10)
math_format()(runif(10))
math_format(format = percent)(runif(10))
```

muted *Mute standard colour.*

Description

Mute standard colour.

Usage

```
muted(colour, l = 30, c = 70)
```

Arguments

colour	character vector of colours to modify
l	new luminance
c	new chroma

Examples

```
muted("red")
muted("blue")
show_col(c("red", "blue", muted("red"), muted("blue")))
```

package-scales *Generic plot scaling methods*

Description

Generic plot scaling methods

parse_format *Parse a text label to produce expressions for plotmath.*

Description

Parse a text label to produce expressions for plotmath.

Usage

```
parse_format()
```

Value

a function with single paramater x, a character vector, that returns a list of expressions

See Also

plotmath

Examples

```
parse_format()(c("alpha", "beta", "gamma"))
```

`percent_format` *Percent formatter: multiply by one hundred and display percent sign.*

Description

Percent formatter: multiply by one hundred and display percent sign.

Usage

```
percent_format()

percent(x)
```

Arguments

x	a numeric vector to format

Value

a function with single paramater x, a numeric vector, that returns a character vector

Examples

```
percent_format()(runif(10))
percent(runif(10))
percent(runif(10, 1, 10))
```

`pretty_breaks` *Pretty breaks. Uses default R break algorithm as implemented in* `pretty`.

Description

Pretty breaks. Uses default R break algorithm as implemented in `pretty`.

Usage

```
pretty_breaks(n = 5, ...)
```

Arguments

n	desired number of breaks
...	other arguments passed on to pretty

Examples

```
pretty_breaks()(1:10)
pretty_breaks()(1:100)
pretty_breaks()(as.Date(c("2008-01-01", "2009-01-01")))
pretty_breaks()(as.Date(c("2008-01-01", "2090-01-01")))
```

---

probability_trans *Probability transformation.*

Description

Probability transformation.

Usage

```
probability_trans(distribution, ...)
```

Arguments

distribution	probability distribution. Should be standard R abbreviation so that "p" + distribution is a valid probability density function, and "q" + distribution is a valid quantile function.
...	other arguments passed on to distribution and quantile functions

---

Range-class *Mutable ranges.*

Description

Mutable ranges have a two methods (train and reset), and make it possible to build up complete ranges with multiple passes.

reciprocal_trans *Reciprocal transformation.*

Description

Reciprocal transformation.

Usage

```
reciprocal_trans()
```

rescale *Rescale numeric vector to have specified minimum and maximum.*

Description

Rescale numeric vector to have specified minimum and maximum.

Usage

```
rescale(x, to = c(0, 1), from = range(x, na.rm = TRUE))
```

Arguments

x	numeric vector of values to manipulate.
to	output range (numeric vector of length two)
from	input range (numeric vector of length two). If not given, is calculated from the range of x

Examples

```
rescale(1:100)
rescale(runif(50))
rescale(1)
```

rescale_max *Rescale numeric vector to have specified maximum.*

Description

Rescale numeric vector to have specified maximum.

Usage

```
rescale_max(x, to = c(0, 1), from = range(x, na.rm = TRUE))
```

Arguments

x	numeric vector of values to manipulate.
to	output range (numeric vector of length two)
from	input range (numeric vector of length two). If not given, is calculated from the range of x

Examples

```
rescale_max(1:100)
rescale_max(runif(50))
rescale_max(1)
```

rescale_mid *Rescale numeric vector to have specified minimum, midpoint, and maximum.*

Description

Rescale numeric vector to have specified minimum, midpoint, and maximum.

Usage

```
rescale_mid(x, to = c(0, 1), from = range(x, na.rm = TRUE), mid = 0)
```

Arguments

x	numeric vector of values to manipulate.
to	output range (numeric vector of length two)
from	input range (numeric vector of length two). If not given, is calculated from the range of x
mid	mid-point of input range

Examples

```
rescale_mid(1:100, mid = 50.5)
rescale_mid(runif(50), mid = 0.5)
rescale_mid(1)
```

`rescale_none` *Don't peform rescaling*

Description

Don't peform rescaling

Usage

```
rescale_none(x, ...)
```

Arguments

`x`	numeric vector of values to manipulate.
`...`	all other arguments ignored

Examples

```
rescale_none(1:100)
```

`rescale_pal` *Rescale palette (continuous).*

Description

Just rescales the input to the specific output range. Useful for alpha, size, and continuous position.

Usage

```
rescale_pal(range = c(0.1, 1))
```

Arguments

`range`	Numeric vector of length two, giving range of possible values. Should be between 0 and 1.

reverse_trans *Reverse transformation.*

Description

Reverse transformation.

Usage

```
reverse_trans()
```

scientific_format *Scientific formatter.*

Description

Scientific formatter.

Usage

```
scientific_format(digits = 3, ...)

scientific(x, digits = 3, ...)
```

Arguments

digits	number of significant digits to show
...	other arguments passed on to format
x	a numeric vector to format

Value

a function with single paramater x, a numeric vector, that returns a character vector

Examples

```
scientific_format()(1:10)
scientific_format()(runif(10))
scientific_format(digits = 2)(runif(10))
scientific(1:10)
scientific(runif(10))
scientific(runif(10), digits = 2)
```

seq_gradient_pal *Sequential colour gradient palette (continous).*

Description

Sequential colour gradient palette (continous).

Usage

```
seq_gradient_pal(low = mnsl("10B 4/6"), high = mnsl("10R 4/6"),
  space = "Lab")
```

Arguments

low	colour for low end of gradient.
high	colour for high end of gradient.
space	colour space in which to calculate gradient. "Lab" usually best unless gradient goes through white.

Examples

```
x <- seq(0, 1, length = 25)
show_col(seq_gradient_pal()(x))
show_col(seq_gradient_pal("white", "black")(x))

library(munsell)
show_col(seq_gradient_pal("white", mnsl("10R 4/6"))(x))
```

shape_pal *Shape palette (discrete).*

Description

Shape palette (discrete).

Usage

```
shape_pal(solid = TRUE)
```

Arguments

solid	should shapes be solid or not?

show_col *Show colours.*

Description

A quick and dirty way to show colours in a plot.

Usage

```
show_col(colours)
```

Arguments

colours	a character vector of colours

sqrt_trans *Square-root transformation.*

Description

Square-root transformation.

Usage

```
sqrt_trans()
```

squish *Squish values into range.*

Description

Squish values into range.

Usage

```
squish(x, range = c(0, 1), only.finite = TRUE)
```

Arguments

x	numeric vector of values to manipulate.
range	numeric vector of length two giving desired output range.
only.finite	if TRUE (the default), will only modify finite values.

Author(s)

Homer Strong <homer.strong@gmail.com>

Examples

```
squish(c(-1, 0.5, 1, 2, NA))
squish(c(-1, 0, 0.5, 1, 2))
```

---

squish_infinite *Squish infinite values to range.*

Description

Squish infinite values to range.

Usage

```
squish_infinite(x, range = c(0, 1))
```

Arguments

x	numeric vector of values to manipulate.
range	numeric vector of length two giving desired output range.

Examples

```
squish_infinite(c(-Inf, -1, 0, 1, 2, Inf))
```

---

time_trans *Transformation for times (class POSIXt).*

Description

Transformation for times (class POSIXt).

Usage

```
time_trans(tz = NULL)
```

Arguments

tz	Optionally supply the time zone. If NULL, the default, the time zone will be extracted from first input with a non-null tz.

Examples

```
hours <- seq(ISOdate(2000,3,20, tz = ""), by = "hour", length.out = 10)
t <- time_trans()
t$trans(hours)
t$inv(t$trans(hours))
t$format(t$breaks(range(hours)))
```

---

trans_breaks *Pretty breaks on transformed scale.*

Description

These often do not produce very attractive breaks.

Usage

```
trans_breaks(trans, inv, n = 5, ...)
```

Arguments

trans	function of single variable, x, that given a numeric vector returns the transformed values
inv	inverse of the transformation function
n	desired number of ticks
...	other arguments passed on to pretty

Examples

```
trans_breaks("log10", function(x) 10 ^ x)(c(1, 1e6))
trans_breaks("sqrt", function(x) x ^ 2)(c(1, 100))
trans_breaks(function(x) 1 / x, function(x) 1 / x)(c(1, 100))
trans_breaks(function(x) -x, function(x) -x)(c(1, 100))
```

---

trans_format *Format labels after transformation.*

Description

Format labels after transformation.

Usage

```
trans_format(trans, format = scientific_format())
```

Arguments

trans	transformation to apply
format	additional formatter to apply after transformation

Value

a function with single paramater x, a numeric vector, that returns a character vector of list of expressions

Examples

```
tf <- trans_format("log10", scientific_format())
tf(10 ^ 1:6)
```

---

trans_new *Create a new transformation object.*

Description

A transformation encapsulates a transformation and its inverse, as well as the information needed to create pleasing breaks and labels. The breaks function is applied on the transformed range of the range, and it's expected that the labels function will perform some kind of inverse tranformation on these breaks to give them labels that are meaningful on the original scale.

Usage

```
trans_new(name, transform, inverse, breaks = extended_breaks(),
  format = format_format(), domain = c(-Inf, Inf))
```

Arguments

name	transformation name
transform	function, or name of function, that performs the transformation
inverse	function, or name of function, that performs the inverse of the transformation
breaks	default breaks function for this transformation. The breaks function is applied to the raw data.
format	default format for this transformation. The format is applied to breaks generated to the raw data.
domain	domain, as numeric vector of length 2, over which transformation is valued

See Also

asn_trans, atanh_trans, boxcox_trans, date_trans, exp_trans, identity_trans, log10_trans, log1p_trans, log2_trans, log_trans, logit_trans, probability_trans, probit_trans, reciprocal_trans, reverse_trans, sqrt_trans, time_trans

trans_range *Compute range of transformed values.*

Description

Silently drops any ranges outside of the domain of `trans`.

Usage

```
trans_range(trans, x)
```

Arguments

`trans`	a transformation object, or the name of a transformation object given as a string.
`x`	a numeric vector to compute the rande of

zero_range *Determine if range of vector is close to zero, with a specified tolerance*

Description

The machine epsilon is the difference between 1.0 and the next number that can be represented by the machine. By default, this function uses epsilon * 100 as the tolerance. First it scales the values so that they have a mean of 1, and then it checks if the difference between them is larger than the tolerance.

Usage

```
zero_range(x, tol = .Machine$double.eps * 100)
```

Arguments

`x`	numeric range: vector of length 2
`tol`	A value specifying the tolerance. Defaults to `.Machine$double.eps * 100`.

Value

logical `TRUE` if the relative difference of the endpoints of the range are not distinguishable from 0.

Examples

```
eps <- .Machine$double.eps
zero_range(c(1, 1 + eps))       # TRUE
zero_range(c(1, 1 + 99 * eps))  # TRUE
zero_range(c(1, 1 + 101 * eps)) # FALSE - Crossed the tol threshold
zero_range(c(1, 1 + 2 * eps), tol = eps) # FALSE - Changed tol

# Scaling up or down all the values has no effect since the values
# are rescaled to 1 before checking against tol
zero_range(100000 * c(1, 1 + eps))        # TRUE
zero_range(100000 * c(1, 1 + 200 * eps))  # FALSE
zero_range(.00001 * c(1, 1 + eps))        # TRUE
zero_range(.00001 * c(1, 1 + 200 * eps))  # FALSE

# NA values
zero_range(c(1, NA))   # NA
zero_range(c(1, NaN))  # NA

# Infinite values
zero_range(c(1, Inf))     # FALSE
zero_range(c(-Inf, Inf))  # FALSE
zero_range(c(Inf, Inf))   # TRUE
```

Index