Package ‘numform’

October 13, 2022

Title Tools to Format Numbers for Publication

Version 0.7.0

Maintainer Tyler Rinker <tyler.rinker@gmail.com>

Description Format numbers and plots for publication; includes the removal of leading zeros, standardization of number of digits, addition of affixes, and a p-value formatter. These tools combine the functionality of several 'base' functions such as 'paste()', 'format()', and 'sprintf()' into specific use case functions that are named in a way that is consistent with usage, making their names easy to remember and easy to deploy.

Depends R (>= 3.2.0)

Suggests testthat

Imports glue

License GPL-2

URL https://github.com/trinker/numform

BugReports https://github.com/trinker/numform/issues

RoxygenNote 7.1.2

Collate 'alignment.R' 'as_factor.R' 'constants.R' 'f_12_hour.R'
 'utils.R' 'f_abbreviation.R' 'f_affirm.R' 'f_affix.R' 'f_bin.R'
 'f_comma.R' 'f_data.R' 'f_date.R' 'f_degree.R' 'f_denom.R'
 'f_dollar.R' 'f_list.R' 'f_logical.R' 'f_month.R' 'f_num.R'
 'f_ordinal.R' 'f_pad_zero.R' 'f_parenthesis.R' 'f_percent.R'
 'f_pval.R' 'f_quarter.R' 'f_replace.R' 'f_sign.R' 'f_state.R'
 'f_text_bar.R' 'f_title.R' 'f_weekday.R' 'f_wrap.R' 'f_year.R'
 'fv_num_percent.R' 'fv_percent.R' 'fv_percent_diff.R'
 'fv_percent_lead.R' 'fv_runs.R' 'glue-reexports.R'
 'highlight_cells.R' 'numform-package.R' 'round.R'
 'time_digits.R'

NeedsCompilation no

Author Tyler Rinker [aut, cre]

Repository CRAN

Date/Publication 2021-10-09 21:10:12 UTC
R topics documented:

alignment .................................................... 3
as_factor ..................................................... 4
constant_months ............................................ 6
fv_num_percent ............................................. 7
fv_percent ................................................... 8
fv_percent_diff ............................................. 9
fv_percent_lead ........................................... 10
fv_runs ....................................................... 11
f_12_hour ..................................................... 12
f_abbreviation ............................................. 14
f_affirm ....................................................... 14
f_affix ......................................................... 15
f_bin .......................................................... 16
f_comma ....................................................... 20
f_data ........................................................ 21
f_data_abbreviation ...................................... 26
f_date ........................................................ 27
f_denom ....................................................... 27
f_dollar ....................................................... 32
f_fahrenheit ............................................... 33
f_list ........................................................ 37
f_logical ..................................................... 38
f_month ....................................................... 39
f_num ........................................................ 42
f_ordinal ..................................................... 43
f_pad_zero ................................................... 44
f_parenthesis .............................................. 45
f_percent ..................................................... 47
f_pval ........................................................ 49
f_quarter ..................................................... 50
f_replace .................................................... 51
f_sign ........................................................ 52
f_state ....................................................... 53
f_text_bar ................................................... 53
f_title ........................................................ 55
f_weekday .................................................... 57
f_wrap ......................................................... 60
f_year ........................................................ 62
highlight_cells ............................................. 63
numform ...................................................... 65
round2 ....................................................... 65
time_digits .................................................. 66

Index 68
alignment

Detect Column Alignment

Description

Many of the specialized functions in numform can change the type of the data from numeric to character causing the table formatting functions in various add-on packages to improperly align the elements. This function passes the columns with a regular expression to detect alignment regardless of column class.

Usage

alignment(
  x,
  left = "left",
  right = ifelse(left == "l", "r", "right"),
  additional.numeric = paste0("^((<b>(&ndash;|\+)</b>)|(<?(([0-9.-%-]+)|\$?\s*\d+[KBM])))|(NaN|NA|Inf)$",
    "|((\$?\s*\d+[KBM]))|((Na|NaN|NA|Inf)$),"
  sep = NULL,
  ...
)

Arguments

x A data.frame.
left A value to print for left aligned columns.
right A value to print for right aligned columns. If left = "l" right will default to "r" otherwise defaults to "right".
additional.numeric An additional regex to consider as numeric. To turn off this feature use additional.numeric = NULL.
sep A string to collapse the vector on.
... ignored.

Value

Returns a vector of lefts and rights or a string (if sep is not NULL).

Examples

CO <- CO2
CO[] <- lapply(CO, as.character)
alignment(CO)
head(CO2)
## Not run:
library(dplyr)
library(pander)
library(xtable)

set.seed(10)
dat <- data_frame(
  Team = rep(c("West Coast", "East Coast"), each = 4),
  Year = rep(2012:2015, 2),
  YearStart = round(rnorm(8, 2e6, 1e6) + sample(1:10/100, 8, TRUE), 2),
  Won = round(rnorm(8, 4e5, 2e5) + sample(1:10/100, 8, TRUE), 2),
  Lost = round(rnorm(8, 4.4e5, 2e5) + sample(1:10/100, 8, TRUE), 2),
  WinLossRate = Won/Lost,
  PropWon = Won/YearStart,
  PropLost = Lost/YearStart
)
dat %>%
group_by(Team) %>%
  mutate(
    `%\Delta`WinLoss = fv_percent_diff(WinLossRate, 0),
    `%\Delta`WinLoss = f_sign(Won - Lost, `<b>+</b>`, `<b>&ndash;</b>`)  
) %>%
  ungroup() %>%
  mutate_at(vars(Won:Lost), .funs = ff_denom(relative = -1, prefix = `$`)) %>%
  mutate_at(vars(PropWon, PropLost), .funs = ff_prop2percent(digits = 0)) %>%
  mutate(
    YearStart = f_denom(YearStart, 1, prefix = `$`),
    Team = fv_runs(Team),
    WinLossRate = f_num(WinLossRate, 1)
) %>%
as.data.frame() %>%
pander::pander(split.tables = Inf, justify = alignment(.))

alignment(CO, 'l', 'r')

CO %>%
xtable(align = c('', alignment(CO, 'l', 'r'))) %>%
print(include.rownames = FALSE)

CO %>%
xtable(align = c('', alignment(CO, 'l', 'r'))) %>%
print(include.rownames = FALSE)

## End(Not run)
as_factor  

Convert Select numform Outputs to Factor

Description

Convert month and weekday and weekday types to factor with correctly ordered levels. Note that the 'forcats' package imported by the 'tidyverse' package, has an as_factor function that can compete with numform’s version. If in doubt, prefix with numform::as_factor.

Usage

as_factor(x, shift = 0, ...)

Arguments

x    A vector of weekdays or months.
shift Shift the levels to the right or left. Useful for setting the week beginning to something besides Sunday. Use -1 to set to Monday instead.
...  ignored.

Value

Returns a factor vector with levels set.

Examples


## Note that the 'forcats' package imported by the 'tidyverse' package, has an
## 'as_factor' function that can compete with numform's version. If in doubt
## prefix with 'numform::as_factor'

as_factor(dat$month1)
as_factor(dat$month2)
as_factor(dat$weekday1)
as_factor(dat$weekday2)
as_factor(dat$weekday3)
as_factor(dat$weekday4)

## shift levels
as_factor(dat$weekday4, -1)
as_factor(dat$weekday4, -2)
as_factor(dat$weekday4, 1)
as_factor(dat$weekday4, 2)

## Not run:
library(tidyverse)
data_frame(
  revenue = rnorm(10000, 500000, 50000),
  date = sample(seq(as.Date('1999/01/01'), as.Date('2000/01/01'), by="day"), 10000, TRUE),
  site = sample(paste("Site", 1:5), 10000, TRUE)
) %>%
  mutate(
    dollar = f_comma(f_dollar(revenue, digits = -3)),
    thou = f_thous(revenue),
    thou_dollars = f_thous(revenue, prefix = '$'),
    abb_month = f_month(date),
    abb_week = numform::as_factor(f_weekday(date, distinct = TRUE)))
) %>%
print() %>%
ggplot(aes(abb_week, revenue)) +
  geom_jitter(width = .2, height = 0, alpha = .2) +
  scale_y_continuous(label = ff_thous(prefix = '$')) +
  facet_wrap(~site) +
  theme_bw()

## End(Not run)

<table>
<thead>
<tr>
<th>constant_months</th>
<th>Constants</th>
</tr>
</thead>
<tbody>
<tr>
<td>constant_months</td>
<td>A constant for ordered month names.</td>
</tr>
<tr>
<td>constant_months_abbreviation</td>
<td>A constant for ordered month abbreviations.</td>
</tr>
<tr>
<td>constant_weekdays</td>
<td>A constant for ordered weekdays.</td>
</tr>
<tr>
<td>constant_quarters</td>
<td>A constant for ordered quarters.</td>
</tr>
</tbody>
</table>

**Description**

- **constant_months**: A constant for ordered month names.
- **constant_months_abbreviation**: A constant for ordered month abbreviations.
- **constant_weekdays**: A constant for ordered weekdays.
- **constant_quarters**: A constant for ordered quarters.

**Usage**

- `constant_months`
- `constant_months_abbreviation`
- `constant_weekdays`
- `constant_weekdays_abbreviation`
- `constant_quarters`
fv_num_percent

Format
An object of class character of length 12.
An object of class character of length 12.
An object of class character of length 7.
An object of class character of length 7.
An object of class character of length 4.

fv_num_percent (Convert a Numeric Vector to Number and Parenthetical Percentages)

Description
Convert a vector of numbers into a vector of strings with the number followed by the relative percentage in parenthesis.

Usage
fv_num_percent(
  x,
  x_digits =getOption("numformdigits"),
  y_digits = x_digits,
  sep = "",
  comma = TRUE,
  ...
)

ffv_num_percent(...)  

Arguments
x A numeric vector.
x_digits The number of digits to round the x vector.
y_digits The number of digits to round the y vector.
sep The separator between the first number and the leading parenthesis.
comma logical. If TRUE the leading number is comma separated.
... ignored.

Value
Returns a vector of parenthesis combined strings using vector x followed by the value as a relative percent in parenthesis.

Examples
fv_num_percent(1:10)  
fv_num_percent(1:10, x_digits = 0, y_digits = 1, sep = " ")
Convert a Numeric Vector to Percentages

Description

Converts a numeric vector into a vector of relative percentages.

Usage

fv_percent(x, digits = getOption("numformdigits"), ...)

ffv_percent(...)

ffv_percent(...)

Arguments

x

A numeric vector.

digits

The number of digits to use. Defaults to 1. Can be set globally via: options(numformdigits = n) where n is the number of digits beyond the decimal point to include.

...

Other arguments passed to f_prop2percent.

Value

Returns a string of publication ready relative percentages.

Examples

fv_percent(1:4)
fv_percent(sample(1:200, 20))
## Not run:
library(tidyverse)

mtcars %>%
count(cyl, gear) %>%
group_by(cyl) %>%
mutate(perc = fv_percent(n, digits = 0))

mtcars %>%
count(cyl, gear) %>%
group_by(cyl) %>%
mutate(perc = fv_percent(n, digits = 0)) %>%
ggplot(aes(gear, n)) +
  geom_bar(stat = 'identity') +
  facet_wrap(~cyl, ncol = 1) +
  geom_text(aes(y = n + 1, label = perc))

## End(Not run)
fv_percent_diff

Structure

fv_percent_diff - Convert a vector of values to percent differences (i.e., (T2 - T1)/T1).

Usage

fv_percent_diff(x, digits = getOption("numformdigits"), ...)

fv_percent_diff_fixed_relative(
  x,
  fixed.relative = 1,
  digits = getOption("numformdigits"),
  ...)

ffv_percent_diff_fixed_relative(...)

ffv_percent_diff(...)

Arguments

x A numeric vector.
digits The number of digits to use. Defaults to 1. Can be set globally via: options(numformdigits = n) where n is the number of digits beyond the decimal point to include.
fixed.relative The position of the element to be used for comparison. Default is the first element.
... Other arguments passed to f_prop2percent.

Value

Returns a string of publication ready relative percent differences.

Examples

set.seed(10)
x <- sample(1:10)
data.frame(
  original = x,
  perc_change = fv_percent_diff(x))
CO2 %>%
group_by(Plant) %>%
mutate(
  'Percent'  = fv_percent(conc),
  'Percent Diff'  = fv_percent_diff(conc)
) %>%
print(n=Inf)

CO2 %>%
group_by(Type, Treatment) %>%
mutate(
  'Percent'  = fv_percent(conc),
  'Percent Diff'  = fv_percent_diff(conc)
) %>%
print(n=Inf)

## End(Not run)

<table>
<thead>
<tr>
<th>fv_percent_lead</th>
<th>Percent Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>fv_percent_lead - Convert a vector of values to percent relative to prior value in the vector (i.e., T2/T1).</td>
</tr>
<tr>
<td>Usage</td>
<td>fv_percent_lead(x, digits = getOption(&quot;numformdigits&quot;), ...)</td>
</tr>
<tr>
<td></td>
<td>fv_percent_lead_fixed_relative(</td>
</tr>
<tr>
<td></td>
<td>x,</td>
</tr>
<tr>
<td></td>
<td>fixed.relative = 1,</td>
</tr>
<tr>
<td></td>
<td>digits = getOption(&quot;numformdigits&quot;),</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>)</td>
</tr>
<tr>
<td></td>
<td>ffv_percent_lead(...)</td>
</tr>
<tr>
<td></td>
<td>ffv_percent_lead_fixed_relative(...)</td>
</tr>
</tbody>
</table>

Arguments

- **x**: A numeric vector.
- **digits**: The number of digits to use. Defaults to 1. Can be set globally via: options(numformdigits = n) where n is the number of digits beyond the decimal point to include.
fv_runs

fixed.relative The position of the element to be used for comparison. Default is the first element.

... Other arguments passed to f_prop2percent.

Value

Returns a string of publication ready relative percent differences.

Examples

```r
set.seed(10)
x <- sample(1:10)
data.frame(
  original = x,
  perc_change = fv_percent_lead(x)
)

## Not run:
library(dplyr)
CO2 %>%
group_by(Plant) %>%
mutate(
  'Percent' = fv_percent(conc),
  'Percent Diff' = fv_percent_diff(conc),
  'Percent Relative' = fv_percent_lead(conc)
) %>%
print(n=Inf)

CO2 %>%
group_by(Type, Treatment) %>%
mutate(
  'Percent' = fv_percent(conc),
  'Percent Diff' = fv_percent_diff(conc),
  'Percent Relative' = fv_percent_lead(conc)
) %>%
print(n=Inf)

## End(Not run)
```

fv_runs Remove Subsequent Runs from a Vector

Description

Remove subsequent runs from a vector.
Usage

fv_runs(x, fill = "", missing = NA, ...)

Arguments

x A vector with runs.
fill What to fill in subsequent runs with.
missing What to fill in missing values with.
... ignored.

Value

Returns a vector of strings with subsequent runs removed.

Examples

x <- c(1, 1, 2, 3, 4, 4, 1, 1, 3, 3, NA, 5)
fv_runs(x)
fv_runs(x, fill = '-')
fv_runs(x, fill = '-', missing = 'X')

## Not run:
library(dplyr)
set.seed(10)
data.frame(
  state = sort(sample(state.name[c(1, 5, 9, 12)], 12, TRUE)),
  val = rnorm(12)
) %>%
  mutate(state2 = fv_runs(state))

## End(Not run)

f_12_hour Format 12 Hour Times

Description

Format times to the typical 12 hour

Usage

f_12_hour(x = Sys.time(), format = "%I:%M %p", pad.char = ",", ...)

## Default S3 method:
f_12_hour(x, format = "%I:%M %p", pad.char = ",", ...)

## S3 method for class 'integer'
f_12_hour

f_12_hour(x, format = "%I:%M %p", pad.char = "", ...)  
## S3 method for class 'numeric'  
f_12_hour(x, format = "%I:%M %p", pad.char = "", ...)  
## S3 method for class 'hms'  
f_12_hour(x, format = "%I:%M %p", pad.char = "", ...)  
ff_12_hour(format = "%I:%M %p", pad.char = "", ...)

Arguments

x 
A vector of coercible times.

format 
A character string specifying the time output format.

pad.char 
A character to use for leading padding if lengths of output are unequal.

... 
Other arguments passed to as.POSIXct.

Value

Returns a string of publication ready 12 hour time stamps.

Examples

f_12_hour(Sys.time())
f_12_hour(Sys.time(), pad.char = '0')
f_12_hour(Sys.time(), pad.char = '')
f_12_hour(Sys.time(), '%I:%M:%S %p')
f_12_hour(c(NA, 0:24), '%I %p')
set.seed(10)
times <- as.POSIXct(sample(seq_len(1e4), 12), origin = '1970-01-01')
paste(f_12_hour(range(times)), collapse = ' to ')
## Not run:
library(tidyverse)
set.seed(10)
data_frame(
  time = as.POSIXct(sample(seq_len(1e4), 12), origin = '1970-01-01'),
  val = sample(1:20, length(time), TRUE)
) %>%
mutate(prop = val/sum(val)) %>%
ggplot(aes(time, prop)) +
  geom_line() +
  scale_x_time(labels = ff_12_hour(format = '%I %p')) +
  scale_y_continuous(labels = ff_prop2percent(digits = 0))
## End(Not run)
f_abbreviation  

Abbreviate Strings

Description
A wrapper for abbreviate for abbreviating strings.

Usage
f_abbreviation(x, length = 5, ...)
ff_abbreviation(...)

Arguments
x  A vector of text strings.
length  The minimum length of the abbreviations.
...  Other arguments passed to abbreviate.

Value
Returns a string vector with strings abbreviated.

See Also
abbreviate

Examples
f_abbreviation(state.name)
f_abbreviation('Cool Variable')

f_affirm  

Yes/No Convert Logical/Dummy Code

Description
Coerce logical (TRUE, FALSE) or or dummy coded elements (0/1) to "Yes"/"No" elements. This function is most useful in plot scales.

Usage
f_affirm(x, true = "Yes", false = "No", ...)
ff_affirm(...)
Arguments

- **x**  A vector of logical or dummy integers. This vector will be coerced to logical.
- **true**  A value for TRUE elements.
- **false**  A value for FALSE elements.
- **...**  ignored.

Value

Returns a string of either "Yes" or "No" elements.

See Also

prettyNum

Examples

```r
f_affirm(c(TRUE, TRUE, FALSE))
f_affirm(c(1, 1, 0, 1, 0, 0, NA))
f_affirm(c(1, 0, 2, .3, -3))
f_affirm(rnorm(20) > 0)
f_affirm(rnorm(20) > 0, "A", "B")
```

## Not run:
library(ggplot2)
library(dplyr)

```r
## Without labels
data_frame(dummy = sample(c(TRUE, FALSE), 30, TRUE)) %>%
  count(dummy) %>%
  ggplot(aes(dummy, n)) +
  geom_bar(stat = "identity")

## With labels
data_frame(dummy = sample(c(TRUE, FALSE), 30, TRUE)) %>%
  count(dummy) %>%
  ggplot(aes(dummy, n)) +
  geom_bar(stat = "identity") +
  scale_x_discrete(labels = f_affirm)
```

## End(Not run)

---

**f_affix**

*Add String Affixes*

Description

Convenience function to add affixes to strings (prefixes & suffixes).
Usage

f_affix(x, prefix = "", suffix = "", ...)  
ff_affix(...)  

f_prefix(x, prefix = "$", ...)  
ff_prefix(...)  

f_suffix(x, suffix = "%", ...)  
ff_suffix(...)  

Arguments

x A vector of elements to append with an affix.  
prefix A string to append to the front of elements.  
suffix A string to append to the back of elements.  
... ignored.  

Value

Returns a string of affix appended digits.

Examples

f_affix(1:5, "-", ")  
f_affix(f_num(1:5, 2), "-", ")  

f_prefix(LETTERS[1:5], ")  
f_prefix(f_bills(123456789123, -2), ")  

f_suffix(LETTERS[1:5], ")  
f_suffix(f_num(1:5, 2), ")  

## Not run:  
f_bills(123456789123, -2) %>%  
  f_prefix("")  

## End(Not run)  

f_bin

Convert Binned Intervals to Readable Form
Description

f_bin - Convert binned intervals to symbol form (e.g., "1 < x <= 3").
f_bin_text - Convert binned intervals to text form (e.g., "Greater than or equal to 1 to less than 3").

Usage

f_bin(x, l = "<", le = "<=" , parse = FALSE, ...)

f_bin_text(
  x,
  greater = "Greater than",
  middle = "to",
  less = "less than",
  equal = "or equal to",
  ...
)

f_bin_text_right(x, l = "up to", le = "to", equal.digits = FALSE, ...)

f_bin_right(x, l = "<", le = "<=" , equal.digits = FALSE, parse = FALSE, ...)

ff_bin(l = "<", le = "<=" , parse = TRUE, ...)

ff_bin_text(
  greater = "Greater than",
  middle = "to",
  less = "less than",
  equal = "or equal to",
  ...
)

ff_bin_right(l = "<", le = "<=" , equal.digits = FALSE, parse = TRUE, ...)

f_interval(x, l = "<", le = "<=" , parse = FALSE, ...)

f_interval_text(
  x,
  greater = "Greater than",
  middle = "to",
  less = "less than",
  equal = "or equal to",
  ...
)

f_interval_text_right(x, l = "up to", le = "to", equal.digits = FALSE, ...)

f_interval_right(
  x,
  l = "<",
  le = "\le",
  equal.digits = FALSE,
  parse = FALSE,
  ...
)

ff_interval(l = "<", le = "\le", parse = TRUE, ...)

ff_interval_text(
  greater = "Greater than",
  middle = "to",
  less = "less than",
  equal = "or equal to",
  ...
)

ff_interval_text_right(l = "up to", le = "to", equal.digits = FALSE, ...)

ff_interval_right(l = "<", le = "\le", equal.digits = FALSE, parse = TRUE, ...)

Arguments

x A vector of binned numbers from cut.
1 Less than symbol.
le Less than or equal to symbol.
parse logical. If TRUE is parsed for ggplot2 facet labels.
greater String to use for greater.
middle String to use for middle (defaults to 'to').
less String to use for less.
equal String to use for equal to. This is combined with the less or greater.
equal.digits logical. If TRUE digits are given equal number of decimal places.
... ignored.

Value

f_bin - Returns human readable intervals in symbol form.
f_bin - Returns human readable intervals in word form.
f_bin_text_right - Returns human readable right hand of intervals in word form.
f_bin_right - Returns human readable right hand intervals in symbol form.
Examples

```r
x <- cut(-1:5, 3, right = FALSE)
y <- cut(-4:10, c(-5, 2, 6, 10), right = TRUE)
z <- cut(-4:10, c(-4, 2, 6, 11), right = FALSE)

f_bin(x)
f_interval(x) # `_interval` and `_bin` are interchangeable aliases in the function names
f_bin(y)
f_bin(z)
## HTML
f_bin(z, le = '&le;')

f_bin_text(x)
f_bin_text(y)
f_bin_text(z)
f_bin_text(x, middle = 'but')
f_bin_text(x, greater = 'Above', middle = '', equal = '', less = 'to')
f_bin_text(z, greater = 'From', middle = '', equal = '', less = 'up to')

f_bin_text_right(x)
f_bin_text_right(y)
f_bin_text_right(x, equal.digits = TRUE)
```

```r
## HTML
f_bin_right(y, le = '&le;')

## Not run:
library(tidyverse)

mtcars %>%
  mutate(mpg2 = cut(mpg, 3)) %>%
  ggplot(aes(disp, hp)) +
  geom_point() +
  facet_wrap(~ mpg2,
    labeller = ff_bin() )

mtcars %>%
  mutate(mpg2 = cut(mpg, 3)) %>%
  ggplot(aes(disp, hp)) +
  geom_point() +
  facet_wrap(~ mpg2,
    labeller = function(x) f_bin_right(x, parse = TRUE) )

mtcars %>%
  mutate(mpg2 = cut(mpg, 3, right = FALSE)) %>%
  ggplot(aes(disp, hp)) +
```
f_comma

Comma Format Large Integers

Description

Add commas to larger integers.

Usage

f_comma(x, mark = ",", prefix = "", ...)  
ff_comma(...)  

Arguments

x A vector of numbers (or string equivalents).  
mark The character to include every n places.  
prefix A string to append to the front of elements.  
... Other arguments passed to prettyNum.
Value

Returns a comma separated string of publication ready digits.

See Also

    prettyNum

Examples

    set.seed(4)
    f_comma(sample(4:10, 5)^5)
    f_comma(c(1234.12345, 1234567890, .000034034, 123000000000, -1234567))

---

f_data

Convert and Abbreviate Units of Data.

Description

Convert numeric data to shorter form with unit abbreviations attached. For example, move from
10,000,000,000 (Bytes) to 10GB (Gigabytes) instead.

    f_byte - Force the abbreviation to bytes unit (B).
    f_kilo - Force the abbreviation to kilobytes unit (KB).
    f_mega - Force the abbreviation to megabytes unit (MB).
    f_giga - Force the abbreviation to gigabytes unit (GB).
    f_tera - Force the abbreviation to terabytes unit (TB).
    f_peta - Force the abbreviation to petabytes unit (PB).
    f_exa - Force the abbreviation to exabytes unit (EB).
    f_zetta - Force the abbreviation to zettabytes unit (ZB).
    f_yotta - Force the abbreviation to yottabytes unit (YB).

Usage

    f_data(
        x,
        binary = FALSE,
        digits = 0,
        pad.char = " ",
        less.than.replace = FALSE,
        sep = " ",
        mix.units = FALSE,
        from = "B",
        ...
    )
ff_data(...) 

f_byte(
    x,
    to = "B",
    binary = FALSE,
    digits = 0,
    suffix = f_data_abbreviation(to),
    pad.char = " ",
    less.than.replace = FALSE,
    from = "B",
    sep = " ",
    ...
)

ff_byte(...) 

f_kilo(
    x,
    to = "KB",
    binary = FALSE,
    digits = 0,
    suffix = f_data_abbreviation(to),
    pad.char = " ",
    less.than.replace = FALSE,
    from = "B",
    sep = " ",
    ...
)

ff_kilo(...) 

f_mega(
    x,
    to = "MB",
    binary = FALSE,
    digits = 0,
    suffix = f_data_abbreviation(to),
    pad.char = " ",
    less.than.replace = FALSE,
    from = "B",
    sep = " ",
    ...
)

ff_mega(...) 

f_giga(}
\[ \text{f_data}(x, \text{to} = \"GB\", \text{binary} = \text{FALSE}, \text{digits} = 0, \text{suffix} = \text{f_data_abbreviation}(\text{to}), \text{pad.char} = \"\", \text{less.than.replace} = \text{FALSE}, \text{from} = \"B\", \text{sep} = \"\", \ldots) \]

\[ \text{ff_giga}(\ldots) \]

\[ \text{f_tera}(x, \text{to} = \"TB\", \text{binary} = \text{FALSE}, \text{digits} = 0, \text{suffix} = \text{f_data_abbreviation}(\text{to}), \text{pad.char} = \"\", \text{less.than.replace} = \text{FALSE}, \text{from} = \"B\", \text{sep} = \"\", \ldots) \]

\[ \text{ff_tera}(\ldots) \]

\[ \text{f_peta}(x, \text{to} = \"PB\", \text{binary} = \text{FALSE}, \text{digits} = 0, \text{suffix} = \text{f_data_abbreviation}(\text{to}), \text{pad.char} = \"\", \text{less.than.replace} = \text{FALSE}, \text{from} = \"B\", \text{sep} = \"\", \ldots) \]

\[ \text{ff_peta}(\ldots) \]

\[ \text{f_exa}(x, \text{to} = \"EB\", \text{binary} = \text{FALSE}, \ldots) \]
digits = 0,
suffix = f_data_abbreviation(to),
pad.char = " ",
less.than.replace = FALSE,
from = "B",
sep = " ",
...
)

ff_exa(...)

f_zetta(
  x,
  to = "ZB",
  binary = FALSE,
digits = 0,
suffix = f_data_abbreviation(to),
pad.char = " ",
less.than.replace = FALSE,
from = "B",
sep = " ",
...
)

ff_zetta(...)

f_yotta(
  x,
  to = "YB",
  binary = FALSE,
digits = 0,
suffix = f_data_abbreviation(to),
pad.char = " ",
less.than.replace = FALSE,
from = "B",
sep = " ",
...
)

ff_yotta(...)

Arguments

x A vector of data units.

binary logical. If TRUE the result uses binary conversion, otherwise decimal conversion is used. See https://en.wikipedia.org/wiki/Binary_prefix for additional information on standards.

digits The number of digits to round to.
pad.char

A character to use for leading padding if lengths of output are unequal. Use NA to forgo padding.

less.than.replace

logical. If TRUE values lower than lowest place value will be replaced with a less than sign followed by the integer representation of the place value. For example, if "0GB" then replacement will be "<1GB".

sep

The separator to use between the number and data unit abbreviation.

mix.units

logical. If TRUE then units can be mixed. Typically, this is not a good idea for the sake of comparison. It is most useful when there is a total row which is a sum of the column and this value's unit exceeds the unit of the rest of the column.

from

The starting unit. Typically, this is assumed to be 'Bytes' ('B'). Must be one of c("Bit", "Byte", "Kilobyte", "Megabyte", "Gigabyte", "Terabyte", "Petabyte", "Exabyte", "Zettabyte", "Yottabyte") or c("b", "B", "KB", "MB", "GB", "TB", "PB", "EB", "ZB", "YB"). These are case sensitive.

to

The units to convert to. See the from parameter for accepted units.

suffix

A suffix to use for the units at the end of the numeric string. Typically the user will not interact with this argument. Meant for internal modularity of functions.

... ignored.

Value

Returns a converted and abbreviated vector of units of data.

Examples

## Not run:
x <- c(NA, '3', '- ', -233456789, -2334567890, 10^(0:10))
f_data(x)
f_data(x, pad.char = NA)
f_data(x, mix.units = TRUE)
f_data(x, mix.units = TRUE, binary = TRUE)
f_data(x, mix.units = TRUE, binary = TRUE, digits = 2)
f_byte(100000000, from = 'GB', binary = TRUE)
f_giga(1000000000)
f_giga(10000000000, suffix = 'Gb')

library(tidyverse)
set.seed(15)
dat <- data_frame(
  bytes = round(rnorm(7, 1e7, 7.95e6), 0),
  days = constant_weekdays %>%
    as_factor()
)

dat %>%
mutate(
  data = f_data(bytes, less.than.replace = TRUE),
  weekday = f_weekday(days, distinct = TRUE) %>%
    as_factor()
f_data_abbreviation

Convert Data (byte) Labels to an Abbreviated Form

Description

Convert a data label such as Gigabyte to an abbreviated form like 'GB'.

Usage

f_data_abbreviation(x, ...)

Arguments


... ignored.

Value

Returns a vector of abbreviated data units.

Examples

f_data_abbreviation(x)
f_date

**Format Dates**

**Description**

Format dates to the typical ‘

**Usage**

```r
f_date(x = Sys.Date(), format = "%B %d, %Y", ...) 
```

**Arguments**

- `x` A vector of coercible dates.
- `format` A character string specifying the date output format.
- `...` Other arguments passed to `as.Date`.

**Value**

Returns a string of publication ready dates.

**Examples**

```r
f_date(Sys.Date())
f_date(Sys.time())
f_date(Sys.time(), '%b-%y')
set.seed(10)
dates <- as.Date(sample(1:10000, 12), origin = '1970-01-01')
paste(f_date(range(dates)), collapse = ' to ')
```

---

f_denom

**Abbreviate Numbers**

**Description**

Use the denomination abbreviations K (thousands), M (millions), and B (billions) with abbreviated numbers.

- `f_denom` - Auto-detect the maximum denomination and attempt to use it (if max(x) is < 1K then x is returned).
- `f_trills` - Force the abbreviation to the trillions denomination (B).
- `f_billls` - Force the abbreviation to the billions denomination (B).
- `f_mills` - Force the abbreviation to the millions denomination (B).
- `f_thous` - Force the abbreviation to the thousands denomination (B).
Usage

f_denom(
  x,
  relative = 0,
  prefix = "",
  pad.char = ifelse(prefix == "", NA, " "),
  less.than.replace = FALSE,
  mix.denom = FALSE,
  ...
)

ff_denom(...)

f_trills(
  x,
  relative = 0,
  digits = -12,
  prefix = "",
  pad.char = ifelse(prefix == "", NA, " "),
  less.than.replace = FALSE,
  ...
)

ff_trills(...)

f_bills(
  x,
  relative = 0,
  digits = -9,
  prefix = "",
  pad.char = ifelse(prefix == "", NA, " "),
  less.than.replace = FALSE,
  ...
)

ff_bills(...)

f_mills(
  x,
  relative = 0,
  digits = -6,
  prefix = "",
  pad.char = ifelse(prefix == "", NA, " "),
  less.than.replace = FALSE,
  ...
)

ff_mills(...)
f_thous(
  x,
  relative = 0,
  digits = -3,
  prefix = '',
  pad.char = ifelse(prefix == '', NA, ''),
  less.than.replace = FALSE,
  ...
)

f_denom

Arguments

x A vector of large numbers.

relative A factor relative to the current digits being rounded. For example relative = -1 moves one to the left while relative = 1 moves one to the right.

prefix A string to append to the front of elements.

pad.char A character to use for leading padding if lengths of output are unequal. Use NA to forgo padding.

less.than.replace logical. If TRUE values lower than lowest place value will be replaced with a less than sign followed by the integer representation of the place value. For example, if "$0K" then replacement will be "<1K".

mix.denom logical. If TRUE then denominations can be mixed. Typically this is not a good idea for the sake of comparison. It is most useful when there is a total row which is a sum of the column and this value’s denomination exceeds the denomination of the rest of the column.

digits The number of digits to round to. Actual digits calculated as digits + relative.

Value

Returns an abbreviated vector of numbers.

Examples

f_denom(c(12345, 12563, 191919), prefix = '$')
f_denom(c(12345, 12563, 191919), prefix = '$', pad.char = '')
f_denom(c(1234365, 122123563, 12913919), prefix = '$')
f_denom(c(12343676215, 122126763563, 1291673919), prefix = '$')
f_denom(c(NA, 2, 12343676215, 122126763563, 1291673919), prefix = '$')
f_denom(c(NA, 2, 123436, 122126763, 1291673919), prefix = '$', mix.denom = TRUE)
f_denom(c(NA, 2, 12343676215, 122126763563, 1291673919), prefix = '$', pad.char = '')
f_denom(c(NA, 2, 12343676215, 122126763563, 1291673919), relative = 1, prefix = '$')
f_denom(c(NA, 2, 12343676215, 122126763563, 1291673919), relative = 9, prefix = '$')
f_denom(c(NA, 2, 12343676215, 122126763563, 1291673919), less.than.replace = TRUE)
f_thous(1234)
f_thous(12345)
f_thous(123456)
f_mills(1234567)
f_mills(12345678)
f_mills(123456789)
f_bills(1234567891)
f_bills(12345678912)
f_bills(123456789123)

f_bills(123456789123, -1)  # round to tens
f_bills(123456789123, -2)  # round to hundreds
f_bills(123456789123, +1)  # round to tenths
f_bills(123456789123, +2)  # round to hundreths

x <- c(3886902.8696, 4044584.0424, 6591893.2104, 591893.2104, -3454678)
f_mills(x)
f_mills(x, 1)
f_mills(x, 1, prefix = '$')
f_mills(x, 1, prefix = '$', pad.char = '0')

## Not run:
if (!require("pacman")) install.packages("pacman")
pacman::p_load(tidyverse, magrittr)

f_bills(123456789123, -2) %>%
  f_prefix("$")

data_frame(
  revenue = rnorm(100, 500000, 50000),
  deals = sample(20:50, 100, TRUE)
) %>%
  mutate(
    dollar = f_dollar(revenue, digits = -3),
    thous = f_thous(revenue),
    thous_dollars = f_thous(revenue, prefix = '$')
  ) %>%
  print() %>%
  ggplot(aes(deals, revenue)) +
  geom_point() +
  geom_smooth() +
  scale_y_continuous(label = ff_thous(prefix = '$'))

data_frame(
  revenue = rnorm(10000, 500000, 50000),
  date = sample(seq(as.Date("1999/01/01"), as.Date("2000/01/01"), by="day"), 10000, TRUE),
  site = sample(paste("Site", 1:5), 10000, TRUE)
) %>%
  mutate(
    dollar = f_dollar(revenue, digits = -3),
    thous = f_thous(revenue),
thous_dollars = f_thous(revenue, prefix = '$'),
abb_month = f_month(date),
abb_week = factor(f_weekday(date, distinct = TRUE),
   levels = c('Su', 'M', 'T', 'W', 'Th', 'F', 'S'))
) %>%
print() %>%
ggplot(aes(abb_week, revenue)) +
   geom_jitter(width = .2, height = 0, alpha = .2) +
   scale_y_continuous(label = ff_thous(prefix = '$')) +
   facet_wrap(~site)

set.seed(10)
data_frame(
  w = paste(constant_months, rep(2016:2017, each = 12))[1:20] ,
  x = rnorm(20, 200000, 75000)
) %>%
{
  a <- .
  rbind(
    a,
    a %>%
    mutate(w = 'Total') %>%
    group_by(w) %>%
    summarize(x = sum(x))
  )
} %>%
mutate(
  y = f_denom(x, prefix = '$'),
  z = f_denom(x, mix.denom = TRUE, prefix = '$')
) %>%
data.frame(stringsAsFactors = FALSE, check.names = FALSE) %>%
pander::pander(split.tables = Inf, justify = alignment(.))

## Scale with mixed units
library(tidyverse)
library(numform)

dat <- data_frame(
  Value = c(111, 2345, 34567, 456789, 1000001, 1000000001),
  Time = 1:6)

## Uniform units
ggplot(dat, aes(Time, Value)) +
geom_line() +
scale_y_continuous(labels = ff_denom(prefix = '$'))

## Mixed units
ggplot(dat, aes(Time, Value)) +
geom_line() +
scale_y_continuous(labels = ff_denom(mix.denom = TRUE, prefix = '$', pad.char = ''))

## End(Not run)
f_dollar - A wrapper for f_num that formats dollar values as labeled dollars.

Usage

f_dollar(x, leading_zero = TRUE, digits = 2, p = "$", ...) ff_dollar(...)

Arguments

x A vector of values.
leading_zero logical. If TRUE a leading zero will be added to values from 0 up to 1.
digits The number of digits to use. Defaults to 2. Can be set globally via: options(numformdigits = n) where n is the number of digits beyond the decimal point to include.
p A string to paste at the beginning of the output from f_num. Defaults to dollar sign. This could be useful, for example, to turn a single dollar sign into an escaped version for LaTeX output.
... Other values passed to f_num.

Value

Returns a string of publication ready digits.

See Also

f_num

Examples

f_dollar(c(30, 33.45, .1))
## Not run:
library(dplyr)

f_dollar(c(0.0, 0, .2, -00.02, 1122222, pi)) %>% f_comma()

## End(Not run)
f_fahrenheit

Format Degrees (e.g., Temperature, Coordinates)

Description

Format numbers into degree format for strings, text, titles, and scales.

Usage

f_fahrenheit(
  x,
  digits =getOption("numformdigits"),
  prefix = NULL,
  suffix = TRUE,
  absolute.value = suffix,
  type = "scale",
  symbol = "&deg;",
  ...
)

f_celcius(
  x,
  digits =getOption("numformdigits"),
  prefix = NULL,
  suffix = TRUE,
  absolute.value = suffix,
  type = "scale",
  symbol = "&deg;",
  ...
)

f_longitude(
  x,
  digits =getOption("numformdigits"),
  prefix = NULL,
  suffix = TRUE,
  absolute.value = suffix,
  type = "scale",
  symbol = "&deg;",
  ...
)

f_latitude(
  x,
  digits =getOption("numformdigits"),
  prefix = NULL,
  suffix = TRUE,
absolute.value = suffix,
type = "scale",
symbol = "&deg;",
...
)

f_degree(
  x,
  type = c("scale", "text", "scale", "title", "string"),
  digits = getOption("numformdigits"),
  prefix = NULL,
  suffix = TRUE,
  absolute.value = suffix,
  symbol = "&deg;",
  measure = c("fahrenheit", "celcius", "C", "F", "longitude", "latitude"),
  ...
)

ff_degree(...)

ff_celcius(...)

ff_fahrenheit(...)

ff_longitude(...)

ff_latitude(...)

Arguments

x                  A vector of values.
digits            The number of digits to use. Defaults to 1. Can be set globally via: options(numformdigits = n) where n is the number of digits beyond the decimal point to include.
prefix            A prefix to use before the parenthesis + units when type = 'title'.
suffix            logical. If TRUE a suffix will be added corresponding to the measure:
celcius           A capital C will be used
fahrenheit        A capital F will be used
longitude          Capital W and E will be used
latitude           Capital S and N will be used
absolute.value     logical. If TRUE the absolute value of x will be used. This is useful for coordinates when E/W or N/S indicate direction.
type               One of c('scale', 'text', 'title', 'string'):
scale              To be used for ggplot2 scales (i.e., axis or legend)
text               To be used for ggplot2 text (i.e., geom_text, annotate; note that parse = TRUE must be set
fahrenheit

**title**  To be used for `ggplot2` titles (e.g., main title, axis title, legend title); ignores x values

**string**  To be used for plain text, especially table formatting and allows control over the degree symbol used

**symbol**  A symbol to use for degree when `type = 'string'`.

**measure**  One of c('fahrenheit', 'celcius', 'C', 'F', 'longitude', 'latitude'). There are functions by these names (e.g., `f_celcius`) but not C or F. These functions may be clearer than using `f_degree` and then specifying `measure`.

... ignored.

**Value**

Returns number string(s) with degree symbols.

**Note**

Note that this function differs a bit from other `f_` functions in that in needs a `type`. This is because other `f_` functions return a plain text representation that is generalizable across usages (titles, tables, axis, geom_text, etc). This function has notation that requires special parsing by various usages hence requiring the `type` argument.

**Examples**

```r
## used for ggplot2 axis.text & legend scale
f_celcius(37, type = 'scale')

## used for ggplot2 geom_text
f_celcius(37, type = 'text')

## used for ggplot2 titles
f_celcius(prefix = "My Title", type = 'title')

## used for table and string formatting
f_celcius(37, type = 'string')

f_celcius(37, type = 'string', symbol = '\textdegree')  # LaTeX

## Not run:
if (!require("pacman")) install.packages("pacman")
pacman::p_load(tidyverse, maps, viridis, mapproj)

states <- map_data("state")
arrests <- USArrests
names(arrests) <- tolower(names(arrests))
arrests$region <- tolower(rownames(USArrests))
choro <- merge(states, arrests, sort = FALSE, by = "region")
choro <- choro[order(choro$order), ]

ggplot(choro, aes(long, lat)) +
  geom_polygon(aes(group = group, fill = assault)) +
  coord_map("albers", at0 = 45.5, lat1 = 29.5) +
```
```r
scale_y_continuous(labels = f_latitude) +
scale_x_continuous(labels = f_longitude)

ggplot(choro, aes(long, lat)) +
group+geom_polygon(aes(group = group, fill = assault)) +
coord_map("albers", at0 = 45.5, lat1 = 29.5) +
scale_y_continuous(labels = ff_latitude(suffix = FALSE)) +
scale_x_continuous(labels = ff_longitude(suffix = FALSE))

world <- map_data(map="world")

ggplot(world, aes(map_id = region, x = long, y = lat)) +
geom_map(map = world, aes(map_id = region), fill = "grey40",
colour = "grey70", size = 0.25) +
scale_y_continuous(labels = f_latitude) +
scale_x_continuous(labels = f_longitude)

data_frame(
  Event = c("freezing water", "room temp", "body temp", "steak\'s done",
            "hamburger\'s done", "boiling water"),
  F = c(32, 70, 98.6, 145, 160, 212)
) %>%
  mutate(
    C = (F - 32) * (5/9),
    Event = f_title(Event),
    Event = factor(Event, levels = unique(Event))
  ) %>%
  ggplot(aes(Event, F, fill = F)) +
  geom_col() +
  geom_text(aes(y = F + 4, label = f_fahrenheit(F, digits = 1, type = "text")),
            parse = TRUE, color = "grey60") +
  scale_y_continuous(
    labels = f_fahrenheit, limits = c(0, 220), expand = c(0, 0),
    sec.axis = sec_axis(trans = ~(. - 32) * (5/9), labels = f_celcius,
    name = f_celcius(prefix = "Temperature ", type = "title"))
  ) +
  scale_x_discrete(labels = ff_replace(pattern = "\", replacement = "\n")) +
  scale_fill_viridis(option = "magma", labels = f_fahrenheit, name = NULL) +
  theme_bw() +
  labs(
    y = f_fahrenheit(prefix = "Temperature ", type = "title"),
    title = f_fahrenheit(prefix = "Temperature of Common Events ", type = "title"))
  ) +
  theme(
    axis.ticks.x = element_blank(),
    panel.border = element_rect(fill = NA, color = "grey80"),
    panel.grid.minor.x = element_blank(),
    panel.grid.major.x = element_blank()
  )
```

data_frame(
  Event = c('freezing water', 'room temp', 'body temp', 'steak\'s done',
            'hamburger\'s done', 'boiling water', 'sun surface', 'lighting'),
  F = c(32, 70, 98.6, 145, 160, 212, 9941, 50000)
) %>%
  mutate(
    Event = f_title(Event),
    C = (F - 32) * (5/9)
  ) %>%
  mutate(
    F = f_degree(F, measure = 'F', type = 'string'),
    C = f_degree(C, measure = 'C', type = 'string', zero = '0.0')
  ) %>%
data.frame(stringsAsFactors = FALSE, check.names = FALSE) %>%
pander::pander(split.tables = Inf, justify = alignment(.))

## End(Not run)

---

### f_list

**Format List Series**

**Description**

f_list - Format a vector of elements as a list series (e.g., `c('A', 'B', 'C')` becomes "A, B, and C").

f_list_amp - A ampersand wrapper for f_list with and = 'amp' set by default.

**Usage**

`f_list(x, and = "and", oxford = TRUE, ...)`

`f_list_amp(x, and = "&", oxford = TRUE, ...)`

ff_list(...)

**Arguments**

- **x**
  - A vector of values to turn into a collapsed series.

- **and**
  - The value to use for the 'and'. Commonly 'and' and '&amp;' are used.

- **oxford**
  - logical. If TRUE an oxford comma is used. If you use FALSE you are a monster.

- **...**
  - ignored.

**Value**

Returns a string that is a list series.
Examples

f_list(1)
f_list(1:2)
f_list(1:3)
f_list(1:5)

x <- c("parents", "Lady Gaga", "Humpty Dumpty")
## Three things you love
sprintf('I love my %s.', f_list(x))
## Your parents are lady Gaga & Humpty Dumpty?????
sprintf('I love my %s.', f_list(x, oxford = FALSE))

sprintf('I love my %s.', f_list(x, and = '&'))
sprintf('I love my %s.', f_list_amp(x))

---

### f_logical

**True/False Convert Logical/Dummy Code**

Coerce logical (TRUE, FALSE) or or dummy coded elements (0/1) to "True"/"False" elements. This function is most useful in plot scales.

#### Usage

```r
f_logical(x, true = "True", false = "False", ...)
ff_logical(...)
```

```r
f_response(x, yes = "Yes", no = "No", ...)
ff_response(...)
```

#### Arguments

- **x**
  - A vector of logical or dummy integers. This vector will be coerced to logical.
- **true**
  - A value for TRUE elements.
- **false**
  - A value for FALSE elements.
- **yes**
  - A value for TRUE elements.
- **no**
  - A value for FALSE elements.
- **...**
  - ignored.

#### Value

Returns a string of either "True"/"False" elements.
\textbf{f\_month}

**Description**

Format long month name, integer, or date formats to a single capital letter. Useful for plot scales as a way to save space.

**Usage**

\begin{verbatim}
f_month(x, ...) # Default S3 method: f_month(x, ...) # S3 method for class 'numeric' f_month(x, ...) # S3 method for class 'Date'
\end{verbatim}

\textbf{Examples}

\begin{verbatim}
f_logical(c(TRUE, TRUE, FALSE))
f_logical(c(1, 1, 0, 1, 0, 0, NA))
f_logical(c(1, 0, 2, .3, -3))
f_logical(rnorm(20) > 0)
f_logical(rnorm(20) > 0, "A", "B")

## Not run:
library(ggplot2)
library(dplyr)

## Without labels
data_frame(dummy = sample(c(TRUE, FALSE), 30, TRUE)) %>%
  count(dummy) %>%
  ggplot(aes(dummy, n)) +
  geom_bar(stat = 'identity')

## With labels
data_frame(dummy = sample(c(TRUE, FALSE), 30, TRUE)) %>%
  count(dummy) %>%
  ggplot(aes(dummy, n)) +
  geom_bar(stat = 'identity') +
  scale_x_discrete(labels = f_logical)

## End(Not run)
\end{verbatim}

\textbf{See Also}

\begin{verbatim}
prettyNum
\end{verbatim}
f_month(x, ...)

## S3 method for class 'POSIXt'
 f_month(x, ...)

## S3 method for class 'hms'
 f_month(x, ...)

ff_month(...)

f_month_name(x, ...)

## Default S3 method:
 f_month_name(x, ...)

## S3 method for class 'numeric'
 f_month_name(x, ...)

## S3 method for class 'Date'
 f_month_name(x, ...)

## S3 method for class 'POSIXt'
 f_month_name(x, ...)

## S3 method for class 'hms'
 f_month_name(x, ...)

ff_month_name(...)

f_month_abbreviation(x, ...)

## Default S3 method:
 f_month_abbreviation(x, ...)

## S3 method for class 'numeric'
 f_month_abbreviation(x, ...)

## S3 method for class 'Date'
 f_month_abbreviation(x, ...)

## S3 method for class 'POSIXt'
 f_month_abbreviation(x, ...)

## S3 method for class 'hms'
 f_month_abbreviation(x, ...)

ff_month_abbreviation(....)
Arguments

- **x**: A vector of month names, integers 1-12, or dates.

Value

Returns a single letter month abbreviation atomic vector.

Examples

```r
f_month(month.name)
f_month(1:12)
dates <- seq(as.Date("2000/1/1"), by = "month", length.out = 12)
f_month(dates)
```

```r
set.seed(11)
data_frame(
  date = sample(seq(as.Date("1990/1/1"), by = "day", length.out = 2e4), 12)
) %>%
  mutate(
    year_4 = f_year(date, 2),
    year_2 = f_year(date, 4),
    quarter = f_quarter(date),
    month_name = f_month_name(date) %>%
      as_factor(),
    month_abbreviation = f_month_abbreviation(date) %>%
      as_factor(),
    month_short = f_month(date),
    weekday_name = f_weekday_name(date),
    weekday_abbreviation = f_weekday_abbreviation(date),
    weekday_short = f_weekday(date),
    weekday_short_distinct = f_weekday(date, distinct = TRUE)
)
```

```r
set.seed(10)
dat <- data_frame(
  month = sample(month.name, 1000, TRUE),
  area = sample(LETTERS[1:5], 1000, TRUE)
) %>%
  count(month, area) %>%
  ungroup() %>%
  mutate(month = factor(month, levels = constant_months))
```

```r
## without date formatting
 ggplot(dat, aes(month, n)) +
  geom_bar(stat = 'identity') +
```
```r
facset_wrap(~ area)
## with date formatting
ggplot(dat, aes(month, n)) +
  geom_bar(stat = 'identity') +
  facet_wrap(~ area) +
  scale_x_discrete(labels = f_month)
## End(Not run)
```

---

### f_num

**Format Digits**

**Description**

Remove leading zeros and standardize number of digits. A workhorse for the `numform` package.

**Usage**

```r
f_num(
  x,
  digits = getOption("numformdigits"),
  p,
  s,
  pad.char = NA,
  zero = NULL,
  retain.leading.zero = FALSE,
  ...
)
```

**Arguments**

- `x` A vector of numbers (or string equivalents).
- `digits` The number of digits to use. Defaults to 1. Can be set globally via: `options(numformdigits = n)` where n is the number of digits beyond the decimal point to include.
- `p` A string to paste at the beginning of the output from `f_num`.
- `s` A string to paste at the end of the output from `f_num`.
- `pad.char` A character to use for leading padding if lengths of output are unequal.
- `zero` A value to insert in for zero values.
- `retain.leading.zero` logical. If TRUE then leading zeros before a decimal place are retained.
- `...` ignored.
Value

Returns a string of publication ready digits.

Examples

```r
f_num(c(0.0, 0, .2, -0.02, 1.122222, pi))
f_num(rnorm(10))
f_num(rnorm(20, 100, 200), 0)
f_num(c("-0.23", "0", ".23"))
```

```r
## Percents
f_num(c(30, 33.45, .1), 3, s="%")
```

```r
## Money
f_num(c(30, 33.45, .1), 2, p="$")
```

```r
## Units
f_num(c(30, 33.45, .1), 2, s=" in.<sup>2</sup>")
f_num(c(30, 33.45, .1), 2, p="&Chi;<sup>2</sup>=")
```

```r
## Not run:
library(dplyr)

is.int <- function(x) !all(x %% 1 == 0)

mtcars %>%
  mutate_if(.funs = f_num, is.int)

df <- data.frame(x = -10:10, y = (-10:10)/10)

ggplot(df, aes(x, y)) +
  geom_point() +
  scale_y_continuous(labels = ff_num(zero = 0))
```

```r
## End(Not run)
```

---

*f_ordinal* Add Ordinal Suffixes (-st, -nd, -rd, -th) to Numbers

**Description**

Add ordinal suffixes (-st, -nd, -rd, -th) to numbers.

**Usage**

```r
f_ordinal(x, ...)
```

```r
ff_ordinal(...)
```
Arguments

\begin{itemize}
  \item \texttt{x} \hspace{1cm} A vector of numbers (or string equivalents).
  \item \texttt{...} \hspace{1cm} ignored.
\end{itemize}

Value

Returns a string vector with ordinal suffixes.

Examples

\begin{verbatim}
f_ordinal(1:25)
\end{verbatim}

\begin{verbatim}
f_pad_zero
\end{verbatim}

\begin{verbatim}
Pad Numbers with Leading Zeros
\end{verbatim}

Description

\begin{verbatim}
f_pad_zero\hspace{1cm} - Add leading zeros to numbers.
f_pad_left\hspace{1cm} - Add leading character to strings.
f_pad_right\hspace{1cm} - Add trailing character to strings.
\end{verbatim}

Usage

\begin{verbatim}
f_pad_zero(x, width = NULL, pad.char = "0", ...)
f_pad_left(x, pad.char = " ", width = NULL, ...)
f_pad_right(x, pad.char = " ", width = NULL, ...)
ff_pad_zero(...)  
ff_pad_left(...)  
ff_pad_right(...)  
\end{verbatim}

Arguments

\begin{itemize}
  \item \texttt{x} \hspace{1cm} A vector of numbers (or string equivalents).
  \item \texttt{width} \hspace{1cm} The width to make the strings. Defaults to the maximum number of characters for all elements in \texttt{x}.
  \item \texttt{pad.char} \hspace{1cm} A character to pad the string with.
  \item \texttt{...} \hspace{1cm} ignored.
\end{itemize}

Value

Returns a padded string.
Examples

f_pad_zero(c(NA, 1, 12))
f_pad_zero(c(NA, 1, 100, 10, 1000))
f_pad_zero(as.character(c(NA, 1, 100, 10, 1000)))
f_pad_zero(c(NA, 1, 100, 10, 1000, "B", "BB"))
f_pad_left(c(NA, 1, 100, 10, 1000, "B", "BB"), '-')
f_pad_right(c(NA, 1, 100, 10, 1000, "B", "BB"), '-')
f_pad_left(c(NA, 1, 12))

f_parenthesis  Parenthesis Formatting of Two Vectors

Description

f_parenthesis - Form two vectors of numbers as a leading number followed by a second number in parenthesis.

f_mean_sd - Wrapper for f_parenthesis optimized for formatting vectors of means and standard deviations.

f_num_percent - Wrapper for f_parenthesis optimized for formatting vectors of numbers and percentages deviations.

Usage

f_parenthesis(x, y, sep = "", x_prefix = "", y_prefix = ",", ...)
f_parenthesis(...)

f_mean_sd(x, y, x_digits = 1, y_digits = x_digits, sep = "", ...)
f_mean_sd(...)

f_num_percent(
  x,
  y,
  x_digits = 1,
  y_digits = x_digits,
  sep = ",",
  prop_fun = numform::f_prop2percent,
  ...
)

f_num_percent(...)

ff_num_percent(...)
Arguments

- **x**: Vector 1 (in `f_mean_sd` the mean values and in `f_num_percent` the leading number vector).
- **y**: Vector 2 (in `f_mean_sd` the standard deviation values and in `f_num_percent` the percent/proportion vector).
- **sep**: The separator between the first number and the leading parenthesis.
- **x_prefix**: A constant to place before each value in the x vector.
- **y_prefix**: A constant to place before each value in the y vector inside of the parenthesis.
- **x_digits**: The number of digits to round the x vector.
- **y_digits**: The number of digits to round the y vector.
- **prop_fun**: The proportion function to convert the y vector in `f_num_percent`. Default is `f_prop2percent`. `f_percent` is used for when the values are already percentages.
- **...**: ignored.

Value

Returns a vector of parenthesis combined strings using vector x and y.

Examples

```r
f_parenthesis(
  f_num(sample(50:100, 5), 1),
  f_num(rnorm(5, 5:15, 5), 1),
  prefix = 'mean = ',
  parenthesis_prefix = 'sd = ',
  sep = " "
)

f_mean_sd(rnorm(5, 100, 20), rnorm(5, 20, 5))

f_num_percent(rnorm(5, 100, 20), rnorm(5, .5, .1))

f_parenthesis(
  sample(50:100, 5),
  f_prop2percent(rnorm(5, .5, .1), 0)
)
```

## Not run:
```r
library(tidyverse)
mtcars %>%
group_by(cyl) %>
summarize(
  mean = mean(hp),
  sd = sd(hp),
  n = n()
) %>%
mutate(
```
f_percent

prop = n /sum(n),
mean_sd = f_mean_sd(mean, sd),
n_perc = f_num_percent(n, prop, 0)

## End(Not run)

----

## f_percent

### Format Percentages

**Description**

- `f_percent` - A wrapper for `f_num` that formats percent values as labeled percentages.
- `f_prop2percent` - A wrapper for `f_num` that formats proportions as labeled percentages.
- `f_pp` - A wrapper for `f_prop2percent` that requires less typing and has `digits` set to 0 by default.

### Usage

```r
f_percent(
  x,
  digits = getOption("numformdigits"),
  less.than.replace = FALSE,
  s = "\%",
  ...
)
```

```r
ff_percent(...)```

```r
f_prop2percent(
  x,
  digits = getOption("numformdigits"),
  less.than.replace = FALSE,
  s = "\%",
  ...
)
```

```r
ff_prop2percent(...)```

```r
f_pp(x, digits = 0, less.than.replace = FALSE, s = "\%", ...)
```

```r
ff_pp(...)```

### Arguments

- **x** - A vector of proportions.
- **digits** - The number of digits to use. Defaults to 1. Can be set globally via: `options(numformdigits = n)` where n is the number of digits beyond the decimal point to include.
less.than.replace

logical. If TRUE values lower than lowest place value, specified by digits, will
be replaced with a less than sign followed by the double representation of the
place value specified by digits. For example, if digits = 0 then replacement
will be "<1%" or if digits = 2 then replacement will be "<.01%".

s

A string to paste at the end of the output from f_num. Defaults to percent sign.
This could be useful, for example, to turn a single percent sign into an escaped
version for LaTeX output.

Value

Returns a string of publication ready digits.

See Also

f_num

Examples

f_percent(c(30, 33.45, .1))
f_percent(c(30, 33.45, .1), 1)
f_percent(c(0, 0, .2, -00.02, 1.122222, pi))
f_prop2percent(c(.30, 1, .01, .33, .222, .01))
f_pp(c(.30, 1, .01, .33, .222, .01))

f_percent(c(30, 33.45, .1), digits = 0, less.than.replace = TRUE)
## Escaped for LaTeX:
f_percent(c(30, 33.45, .1), digits = 0, less.than.replace = TRUE, s = '\\%')
f_prop2percent(c(.30, 1, .01, .33, .222, .01, .0001, NA), digits = 0,
  less.than.replace = TRUE)

## Not run:
library(tidyverse)

mtcars %>%
  count(cyl, gear) %>%
  group_by(cyl) %>%
  mutate(prop = n/sum(n)) %>%
  ggplot(aes(gear, prop)) +
  geom_bar(stat = 'identity') +
  facet_wrap(~cyl, ncol = 1) +
  scale_y_continuous(labels = ff_prop2percent(digits = 0))

## End(Not run)
Description

Format p-values for reporting using a < or = sign if greater than alpha level.

Usage

f_pval(
  x,
  alpha = getOption("numformalpha"),
  digits = getOption("numformdigits"),
  ...
)

ff_pval(...)

Arguments

  x  A p-value.

  alpha  The alpha cut off to use. Defaults to .05. Can be set globally via: options(numformalpha = n) where n is the alpha level.

  digits  The number of digits to use. Defaults to 3. Can be set globally via: options(numformdigits = n) where n is the number of digits beyond the decimal point to include.

  ...  Other values passed to f_num.

Value

Returns a string of publication ready p-values.

See Also

  f_num

Examples

  f_pval(.05)
  f_pval(.04999999999999999)
  f_pval(.0002)
  f_pval(.0002, .001)

  mod1 <- t.test(1:10, y = c(7:20))
  f_pval(mod1$p.value)

  mod2 <- t.test(1:10, y = c(7:20, 200))
  f_pval(mod2$p.value)
Description

Format long/abbreviation month name, integer, or date formats to a quarter format (i.e., Q1, Q2, Q3, Q4).

Usage

```r
f_quarter(x, prefix = "Q", space = "", max = 12, ...)  
```

## Default S3 method:
```r
f_quarter(x, prefix = "Q", space = "", max = 12, ...)  
```

## S3 method for class 'numeric'
```r
f_quarter(  
  x,  
  prefix = "Q",  
  space = "",  
  max = ifelse(all(x %in% c(1:4, NA)), 4, 12),  
  ...  
)
```

## S3 method for class 'Date'
```r
f_quarter(x, prefix = "Q", space = "", max = 12, ...)  
```

## S3 method for class 'POSIXt'
```r
f_quarter(x, prefix = "Q", space = "", max = 12, ...)  
```

## S3 method for class 'hms'
```r
f_quarter(x, prefix = "Q", space = "", max = 12, ...)  
```

ff_quarter(prefix = "Q", space = "", max = 12, ...)

Arguments

- `x` A vector of month names, integers 1-12, or dates.
- `prefix` A quarter prefix (defaults to 'Q').
- `space` A string to place between 'Q' and quarter number.
- `max` A maximum in the x vector, if x is numeric, corresponding to months (12) or quarters (4).
- `...` ignored.

Value

Returns a quarter formatted atomic vector.
**Examples**

```r
f_quarter(month.name)

f_quarter(1:12)

dates <- seq(as.Date("2000/1/1"), by = "month", length.out = 12)
f_quarter(dates)
## Not run:
if (!require("pacman")) install.packages("pacman")
pacman::p_load(tidyverse)

set.seed(10)
dat <- data_frame(
  month = sample(month.name, 1000, TRUE),
  area = sample(LETTERS[1:5], 1000, TRUE)
)
mutate(quarter = factor(f_quarter(month), levels = constant_quarters))

## End(Not run)
```

---

**f_replace**

*Replace Characters in Strings*

**Description**

A wrapper for `gsub` for replacing substrings that is useful for `ggplot2` scales. Useful for taking field names like 'Cool_Variable' and turning it into 'Cool Variable'.

**Usage**

```r
f_replace(x, pattern = " ", replacement = " ", ...) 

ff_replace(...) 
```

**Arguments**

- `x` A vector of text strings.
- `pattern` A character string defining search patterns.
- `replacement` A character string defining replacement patterns.
- `...` Other arguments passed to `gsub`.

**Value**

Returns a string vector with characters replaced.
f_sign

Format Numeric Signs

Description
f_sign - Formats numeric values to just their sign ("-" == < 0, '+' == > 0, or "" == 0).

Usage
f_sign(x, positive = "+", negative = "-", zero = "", ...)  
ff_sign(...)

Arguments
x        A vector of values.
positive  A string/value to insert in for positive values.
negative A string/value to insert in for negative values.
zero     A string/value to insert in for zero values.
...      ignored.

Value
Returns a string of signs.

See Also
f_num

Examples
f_sign(c(-10, 0, 10))
f_sign(c(-10, 0, 10), zero = 0)  
## web based
f_sign(c(-10, 0, 10), '<b>++</b>', '<b>&ndash;</b>')
f_state

Format State Names as Abbreviations

Description

Formats a state name as the abbreviated form.

Usage

f_state(x, ...)

ff_state(...)

Arguments

x A vector of states.
... ignored.

Value

Returns a string of abbreviated states.

Examples

f_state(c("Texas", "New York", NA, "New Jersey", "Washington", "Europe"))

f_text_bar

Format Text Based Bar Plots

Description

Use a text symbol to create scaled horizontal bar plots of numeric vectors. Note that you will have to coerce the table to a data.frame in order for the output to look pretty.

Usage

f_text_bar(x, symbol = ",", width = 9, ...)

ff_text_bar(...)

Arguments

x A numeric vector.
symbol A symbol to use for the bars.
width The max width of the bar.
... ignored.
Value

Returns a vector of concatenated symbols as a string that represent x

Examples

```r
## Not run:
library(dplyr)

mtcars %>%
count(cyl, gear) %>%
group_by(cyl) %>%
mutate(
  p = numform::f_pp(n/sum(n))
) %>%
ungroup() %>%
mutate(
  cyl = numform::fv_runs(cyl),
  `\` = f_text_bar(n)  ## Overall
) %>%
as.data.frame()

mtcars %>%
count(cyl, gear) %>%
group_by(cyl) %>%
mutate(
  p = numform::f_pp(n/sum(n)),
  `\` = f_text_bar(n)  ## within groups
) %>%
ungroup() %>%
mutate(
  cyl = numform::fv_runs(cyl),
  `\` = f_text_bar(n)
) %>%
as.data.frame()

mtcars %>%
count(cyl, gear) %>%
group_by(cyl) %>%
mutate(
  p = numform::f_pp(n/sum(n)),
  'within' = f_text_bar(n, width = 30, symbol = '#')
) %>%
ungroup() %>%
mutate(
  cyl = numform::fv_runs(cyl),
  'overall' = f_text_bar(n, width = 30, symbol = '*')
) %>%
as.data.frame() %>%
pander::pander(split.tables = Inf, justify = alignment(.), style = 'simple')
## Drop the headers
mtcars %>%
```
f_title

Convert First Letter of Words to Title Case

Description

A wrapper for toTitleCase converting text to title case.

Usage

f_title(x, upper = NULL, lower = NULL, ...)

ff_title(...)  

Arguments

x A vector of text strings.

upper A vector of regular expression to convert to upper case that would otherwise be lower cased (this should be targeted at the initial output, not the input).

lower A vector of regular expression to convert to lower case that would otherwise be upper cased (this should be targeted at the initial output, not the input).

... ignored.

Value

Returns a string vector with characters replaced.

See Also

toTitleCase
Examples

```r
f_title('i love this title')
f_title(f_replace('Cool Variable'))

f_title(c('select', 'group by', 'My asci'))
f_title(c('select', 'group by', 'My asci'), upper = c('Ascii'))
f_title(c('select', 'group by', 'My asci'), upper = c('Ascii', 'b(?=y\b)'))

## Not run:
library(tidyverse)
set.seed(10)
dat <- data_frame(
  level = c("not_involved", "somewhat_involved_single_group", 
            "somewhat_involved_multiple_groups", "very_involved_one_group", 
            "very_involved_multiple_groups"),
  n = sample(1:10, length(level))
)
mutate(
  level = factor(level, levels = unique(level)),
  '%' = n/sum(n)
)

gridExtra::grid.arrange(
  dat %>%
    ggplot(aes(level, '%')) +
    geom_col() +
    labs(title = 'Very Sad', y = NULL) +
    theme(
      axis.text = element_text(size = 7),
      title = element_text(size = 9)
    ),
  dat %>%
    ggplot(aes(level, '%')) +
    geom_col() +
    scale_x_discrete(lables = function(x) f_replace(x, '_', '\n')) +
    scale_y_continuous(lables = ff_prop2percent(digits = 0)) +
    labs(title = 'Underscore Split (Readable)', y = NULL) +
    theme(
      axis.text = element_text(size = 7),
      title = element_text(size = 9)
    ),
	ncol = 2
),
```
f_weekday

gridExtra::arrangeGrob(

dat %>%
ggplot(aes(level, 'x')) +
geom_col() +
scale_x_discrete(labels = function(x) f_title(f_replace(x))) +
scale_y_continuous(labels = ff_prop2percent(digits = 0)) +
labs(title = 'Underscore Replaced & Title (Capitalized Sadness)', y = NULL) +
theme(
  axis.text = element_text(size = 7),
  title = element_text(size = 9)
),

dat %>%
ggplot(aes(level, 'x')) +
geom_col() +
scale_x_discrete(labels = function(x) f_wrap(f_title(f_replace(x)))) +
scale_y_continuous(labels = ff_prop2percent(digits = 0)) +
labs(title = 'Underscore Replaced, Title, & Wrapped (Happy)', y = NULL) +
theme(
  axis.text = element_text(size = 7),
  title = element_text(size = 9)
),

ncol = 2
), ncol = 1
)

## End(Not run)

f_weekday

Format Weekdays to One Letter Abbreviation

Description

Format long weekday name, integer, or date formats to a single capital letter. Useful for plot scales
as a way to save space.

Usage

f_weekday(x, distinct = FALSE, ...)

# Default S3 method:
f_weekday(x, distinct = FALSE, ...)

# S3 method for class 'numeric'
f_weekday(x, distinct = FALSE, ...)
## S3 method for class 'Date'
f_weekday(x, distinct = FALSE, ...)

## S3 method for class 'POSIXt'
f_weekday(x, distinct = FALSE, ...)

## S3 method for class 'hms'
f_weekday(x, distinct = FALSE, ...)

ff_weekday(distinct = FALSE, ...)

f_weekday_name(x, ...)

## Default S3 method:
f_weekday_name(x, ...)

## S3 method for class 'numeric'
f_weekday_name(x, ...)

## S3 method for class 'Date'
f_weekday_name(x, ...)

## S3 method for class 'POSIXt'
f_weekday_name(x, ...)

## S3 method for class 'hms'
f_weekday_name(x, ...)

ff_weekday_name(...)

f_weekday_abbreviation(x, ...)

## Default S3 method:
f_weekday_abbreviation(x, ...)

## S3 method for class 'numeric'
f_weekday_abbreviation(x, ...)

## S3 method for class 'Date'
f_weekday_abbreviation(x, ...)

## S3 method for class 'POSIXt'
f_weekday_abbreviation(x, ...)

## S3 method for class 'hms'
f_weekday_abbreviation(x, ...)

f_weekday_abbreviation(x, ...)
f_weekday

ff_weekday_abbreviation(...)

Arguments

x A vector of weekday names, integers 1-12, or dates.
distinct logical. If TRUE Sunday will be presented as Su and Thursday as Th.
... ignored.

Value

Returns a single letter month abbreviation atomic vector.

Examples

f_weekday(weekdays(x=as.Date(seq(7), origin="1950-01-07")))
f_weekday(weekdays(x=as.Date(seq(7), origin="1950-01-07")), TRUE)
f_weekday(1:7)
f_weekday(1:7, TRUE)

days <- seq(as.Date("2000/1/2"), by = "day", length.out = 7)
f_weekday(days)
f_weekday(days, TRUE)

## Not run:
if (!require("pacman")) install.packages("pacman")
pacman::p_load(tidyverse)

set.seed(11)
data_frame(
  date = sample(seq(as.Date("1990/1/1"), by = "day", length.out = 2e4), 12)
) %>%
  mutate(
    year_4 = f_year(date, 2),
    year_2 = f_year(date, 4),
    quarter = f_quarter(date),
    month_name = f_month_name(date) %>%
      as_factor(),
    month_abbreviation = f_month_abbreviation(date) %>%
      as_factor(),
    month_short = f_month(date),
    weekday_name = f_weekday_name(date),
    weekday_abbreviation = f_weekday_abbreviation(date),
    weekday_short = f_weekday(date),
    weekday_short_distinct = f_weekday(date, distinct = TRUE)
)

set.seed(10)
dat <- data_frame(
  day = sample(weekdays(days), 10000, TRUE),
  area = sample(LETTERS[1:15], 10000, TRUE)
)
) %>%
  count(day, area) %>%
  ungroup() %>%
  mutate(
    day = factor(day, levels = weekdays(days))
  )

## without date formatting
ggplot(dat, aes(day, n)) +
  geom_bar(stat = "identity") +
  facet_wrap(~area)

## with date formatting
ggplot(dat, aes(day, n)) +
  geom_bar(stat = "identity") +
  facet_wrap(~area) +
  scale_x_discrete(labels = f_weekday)

## with date formatting
ggplot(dat, aes(day, n)) +
  geom_bar(stat = "identity") +
  facet_wrap(~area) +
  scale_x_discrete(labels = ff_weekday(distinct = TRUE))

## End(Not run)

---

**f_wrap**

**Wrap Strings**

**Description**

Wrap strings by splitting n width, and paste collapsing with new line characters.

**Usage**

f_wrap(
  x,
  width = 15,
  sep = "\n",
  exdent = 0,
  indent = 0,
  equal.lines = FALSE,
  collapse = FALSE,
  ...
)

ff_wrap(...)
f_wrap

Arguments

- **x**: A vector of text strings.
- **width**: A positive integer giving the target column for wrapping lines in the output.
- **sep**: A new line separator (defaults to "\n").
- **exdent**: A non-negative integer specifying the indentation of subsequent lines in paragraphs.
- **indent**: A non-negative integer giving the indentation of the first line in a paragraph.
- **equal.lines**: logical. If TRUE the number of lines for each element will be made the same by appending additional 'n' to those below the max number of lines. This is useful for legend spacing.
- **collapse**: logical. If TRUE then x is collapsed via paste(x, collapse = ' ') before processing. This is useful for multi-line text wrapping of longer subtitles.
- **...**: Other arguments passed to strwrap.

Value

Returns a string vector with wrapped new line characters.

See Also

strwrap

Examples

cat(f_wrap('really long label names are the pits'))
cat(f_wrap('really long label names are the pits', width = 20, exdent = 2))
f_wrap(c('really long label names are the pits and make us sad', 'not nearly so long'), equal.lines = TRUE)

## Not run:
library(tidyverse); library(gridExtra)

set.seed(10)
dat <- data_frame(
  level = c('Not Involved', 'Somewhat Involved Single Group', 'Somewhat Involved Multiple Groups', 'Very Involved One Group', 'Very Involved Multiple Groups'),
  n = sample(1:10, length(level))
) %>%
mutate(
  level = factor(level, levels = unique(level)),
  '%' = n/sum(n)
)

gridExtra::grid.arrange(
dat %>%
  ggplot(aes(level, '%')) +
  geom_col() +
  geom_bar(stat = 'identity')
)
f_year

Description
Format 4 digit integer, date, or POSIXlt formats to 2 or 4 digit years.

Usage
f_year(x, digits = 2, ...)

## S3 method for class 'numeric'
f_year(x, digits = 2, ...)

## S3 method for class 'Date'
f_year(x, digits = 2, ...)

## S3 method for class 'POSIXt'
f_year(x, digits = 2, ...)

## S3 method for class 'hms'
f_year(x, digits = 2, ...)

ff_year(digits = 2, ...)

Arguments

x A vector of 4 digits integers, dates, or POSIXlt.
digits Either 2 or 4 for the number of digits to make the year.
... ignored.

Value
Returns a vector of two or four digit years.
**Examples**

```r
f_year(as.Date(paste0(1998:2016, '-12-12')))
f_year(c(NA, 1998:2016, 21345))
## Not run:
library(tidyverse)

dat <- data_frame(
  year = 1998:2016,
  year2 = as.POSIXct(sample(seq_len(1e4), 12), origin = '1970-01-01') +
  (365 * 24 * 3600 * seq_len(19)),
  val = sample(1:20, length(year), TRUE)
) 
%>%
  mutate(prop = val/sum(val))

dat %>%
  ggplot(aes(year, prop)) +
  geom_line() +
  scale_x_continuous(labels = ff_year(digits = 2), breaks = 1998:2016) +
  scale_y_continuous(labels = ff_prop2percent(digits = 0))

dat %>%
  ggplot(aes(year2, prop)) +
  geom_line() +
  scale_x_time(labels = ff_year(digits = 2), breaks = dat$year2) +
  scale_y_continuous(labels = ff_prop2percent(digits = 0))
## End(Not run)
```

**highlight_cells**

A lightweight cell highlighter that uses non-standard evaluation. This function is designed for interactive use. Its behavior outside of this context is not guaranteed. For finer control use an `ifelse` with `paste` within a `?dplyr::mutate` statement.

**Usage**

```r
highlight_cells(
  data,
  rows,
  columns = seq_len(ncol(data)),
  left = "<b>",
  right = gsub("(<)(([^> ]+)([^>]*)>)","\1/\2","", left),
  ...
)
```
### Arguments

- **data**
  A data.frame.

- **rows**
  An expression that evaluates to logical and is equal in length to the number of rows.

- **columns**
  A vector of either integer positions or character names corresponding to columns that should be highlighted. Defaults to all columns.

- **left**
  A highlighting tag for the left side of the cell value.

- **right**
  A highlighting tag for the right side of the cell value. Attempts to use the left input to create a corresponding right HTML based tag.

... ignored.

### Value

Returns a data.frame with the chosen cell values wrapped in highlight tags.

### Examples

```r
highlight_cells(mtcars, rows = hp > 230 | qsec > 20)
highlight_cells(mtcars, rows = hp > 230, columns = 'hp')
```

```r
# Not run:
library(dplyr); library(tibble); library(pander)

mtcars %>%
  highlight_cells(rows = hp > 230, columns = 'hp') %>%
  highlight_cells(rows = qsec > 20, columns = 'qsec', left = '<b style="color:blue;">') %>%
  rownames_to_column('car') %>%
  data.frame(stringsAsFactors = FALSE, check.names = FALSE) %>%
  pander::pander(split.tables = Inf, justify = alignment(.))

# Not run:
if (!require("pacman")) install.packages("pacman")
pacman::p_load(tidyverse, magrittr)

set.seed(10)
data_frame(
  w = paste(constant_months, rep(2016:2017, each = 12))[1:20] ,
  x = rnorm(20, 200000, 75000)
) %>%
  { a <- .
    rbind( 
      a, 
      a %>%
        mutate(w = 'Total') %>%
        group_by(w) %>%
        summarize(x = sum(x))
  )
```
numform

Tools to Format Numbers for Publication

Description

Format numbers and plots for publication; includes the removal of leading zeros, standardization of number of digits, addition of affixes, and a p-value formatter. These tools combine the functionality of several 'base' functions such as `paste`, `format`, and `sprintf` into specific use case functions that are named in a way that is consistent with usage, making their names easy to remember and easy to deploy.

round2

Rounding

Description

round2 - By default R's `round` function uses the 'round half to even' method. This function (taken from https://stackoverflow.com/a/12688836/1000343) rounds half up.

round_any - This tooling lets you round to fractional values, not just whole numbers. Code adapted from https://stackoverflow.com/a/8665247/1000343.

Usage

round2(x, digits = 0, ...)

round_any(x, accuracy, f = round2, ...)

Arguments

x A vector of digits.
digits The number of decimal places to round to.
accuracy Number to round to.
f A function to round (e.g., round, ceiling, floor). Defaults to round2.
... ignored.
**Value**

- `round2` - Returns numeric vector half rounded up.
- `round_any` - Returns a numeric vector or rounded fractional values.

**Author(s)**

Kohske Takahashi

**References**

https://stackoverflow.com/a/12688836/1000343
https://stackoverflow.com/a/8665247/1000343

**Examples**

```r
data.frame(
  orig = .5 + (0:8),
  round = round(.5 + (0:8)),
  round2 = round2(.5 + (0:8))
)
```

```r
round_any(c(.123, 1.234, 4, 4.715), .5)
round_any(c(.123, 1.234, 4, 4.715), .25)
```

---

**time_digits**

Compute Digits Needed for Quarter Hour Time Vector

**Description**

This tool computes the minimum number of digits required for a vector of times. The defaults of the tool assumes your time is rounded to within the quarter hour.

**Usage**

```r
time_digits(x, ...)
```

**Arguments**

- `x` A numeric vector of times rounded to the nearest quarter hour.
- `...` ignored

**Value**

Returns integer 0-2
Examples

time_digits(c(.5, .25, 6))
time_digits(c(.5, 3.5, 6))
time_digits(c(5, 25, 6))

x <- c(.5, .25, 6)
numform::f_pad_left(numform::f_num(x, digits = numform::time_digits(x)))

lapply(
  list(quarter = c(.5, .25, 6), half = c(.5, 3.5, 6), hour = c(5, 25, 6)),
  function(x) numform::f_pad_left(numform::f_num(x, digits = numform::time_digits(x)))
)
Index

 datasets
 constant_months, 6
 abbreviate, 14
 alignment, 3
 as.Date, 27
 as.POSIXct, 13
 as_factor, 4
 constant_months, 6
 constant_months_abbreviation (constant_months), 6
 constant_quarters (constant_months), 6
 constant_weekdays (constant_months), 6
 constant_weekdays_abbreviation (constant_months), 6
 constant_weekdays_abbreviation (constant_months), 6

 f_12_hour, 12
 f_abbreviation, 14
 f_affirm, 14
 f_affix, 15
 f_bills (f_denom), 27
 f_bin, 16
 f_bin_right (f_bin), 16
 f_bin_text (f_bin), 16
 f_bin_text_right (f_bin), 16
 f_byte (f_data), 21
 f_celcius (f_fahrenheit), 33
 f_comma, 20
 f_data, 21
 f_data_abbreviation, 26
 f_date, 27
 f_degree (f_fahrenheit), 33
 f_denom, 27
 f_dollar, 32
 f_exa (f_data), 21
 f_fahrenheit, 33
 f_giga (f_data), 21
 f_interval (f_bin), 16
 f_interval_right (f_bin), 16
 f_interval_text (f_bin), 16
 f_interval_text_right (f_bin), 16
 f_kilo (f_data), 21
 f_latitude (f_fahrenheit), 33
 f_list, 37
 f_list_amp (f_list), 37
 f_logical, 38
 f_longitude (f_fahrenheit), 33
 f_mean_sd (f_parenthesis), 45
 f_mega (f_data), 21
 f_mills (f_denom), 27
 f_month, 39
 f_month_abbreviation (f_month), 39
 f_month_name (f_month), 39
 f_num, 32, 42, 47–49, 52
 f_num_percent (f_parenthesis), 45
 fOrdinal, 43
 f_pad_left (f_pad_zero), 44
 f_pad_right (f_pad_zero), 44
 f_pad_zero, 44
 f_parenthesis, 45
 f_percent, 47
 f_peta (f_data), 21
 f_pp (f_percent), 47
 f_prefix (f_affix), 15
 f_prop2percent, 8, 9, 11, 47
 f_prop2percent (f_percent), 47
 f_pval, 49
 f_quarter, 50
 f_replace, 51
 f_response (f_logical), 38
 f_sign, 52
 f_state, 53
 f_suffix (f_affix), 15
 f_tera (f_data), 21
 f_text_bar, 53
 f_thous (f_denom), 27
INDEX

f-title, 55
f-trills (f-denom), 27
f-weekday, 57
f-weekday-abbreviation (f-weekday), 57
f-weekday-name (f-weekday), 57
f-wrap, 60
f-year, 62
f-yotta (f-data), 21
f-zetta (f-data), 21
ff-12-hour (f-12-hour), 12
ff-abbreviation (f-abbreviation), 14
ff-affirm (f-affirm), 14
ff-affix (f-affix), 15
ff-bills (f-denom), 27
ff-bin (f-bin), 16
ff-bin-right (f-bin), 16
ff-bin-text (f-bin), 16
ff-bin-text-right (f-bin), 16
ff-byte (f-data), 21
ff-celsius (f-fahrenheit), 33
ff-comma (f-comma), 20
ff-data (f-data), 21
ff-date (f-date), 27
ff-degree (f-fahrenheit), 33
ff-denom (f-denom), 27
ff-dollar (f-dollar), 32
ff-exa (f-data), 21
ff-fahrenheit (f-fahrenheit), 33
ff-giga (f-data), 21
ff-interval (f-bin), 16
ff-interval-right (f-bin), 16
ff-interval-text (f-bin), 16
ff-interval-text-right (f-bin), 16
ff-kilo (f-data), 21
ff-latitude (f-fahrenheit), 33
ff-list (f-list), 37
ff-logical (f-logical), 38
ff-longitude (f-fahrenheit), 33
ff-mean_sd (f-parenthesis), 45
ff-mega (f-data), 21
ff-mills (f-denom), 27
ff-month (f-month), 39
ff-month-abbreviation (f-month), 39
ff-month-name (f-month), 39
ff-num (f-num), 42
ff-num-percent (f-parenthesis), 45
ff-ordinal (f-ordinal), 43
ff-pad-left (f-pad-zero), 44
ff-pad-right (f-pad-zero), 44
ff-pad-zero (f-pad-zero), 44
ff-parenthesis (f-parenthesis), 45
ff-percent (f-percent), 47
ff-peta (f-data), 21
ff-pp (f-percent), 47
ff-prefix (f-affix), 15
ff-prop2percent (f-percent), 47
ff-pval (f-pval), 49
ff-quarter (f-quarter), 50
ff-replace (f-replace), 51
ff-response (f-logical), 38
ff-sign (f-sign), 52
ff-state (f-state), 53
ff-suffix (f-affix), 15
ff-era (f-data), 21
ff-text-bar (f-text-bar), 53
ff-thous (f-denom), 27
ff-title (f-title), 55
ff-trills (f-title), 55
ff-weekday (f-weekday), 57
ff-weekday-abbreviation (f-weekday), 57
ff-weekday-name (f-weekday), 57
ff-wrap (f-wrap), 60
ff-year (f-year), 62
ff-yotta (f-data), 21
ff-zetta (f-data), 21
ff-numeric_percent (f-numeric_percent), 7
ff-percent (f-numeric_percent), 8
ff-percent_diff (f-numeric_percent), 9
ff-percent_diff_fixed_relative (f-numeric_percent), 9
ff-percent_lead (f-numeric_percent), 10
ff-percent_lead_fixed_relative (f-numeric_percent), 10
format, 65
fv-numeric_percent, 7
fv-percent, 8
fv-percent_diff, 9
fv-percent_diff_fixed_relative (f-numeric_percent), 9
fv-percent_lead, 10
fv-percent_lead_fixed_relative (f-numeric_percent), 10
fv-runs, 11
gsub, 51
highlight_cells, 63
numform, 65

package-numform (numform), 65
paste, 65
prettyNum, 15, 20, 21, 39

round2, 65
round_any (round2), 65

sprintf, 65
strwrap, 52, 61

time_digits, 66
toTitleCase, 55