iemisc: Construction Measurement Examples

Irucka Embry, E.I.T. (EcoC²S)

2023-09-24

Contents

Replicate the R code

Note: If you wish to replicate the R code below, then you will need to copy and paste the following commands in R first (to make sure you have all the packages and their dependencies):

```
install.packages(c("install.load", "iemisc", "units", "knitr"))
# install the packages and their dependencies



# load the required packages
install.load::load_package("iemisc", "units")
# load needed packages using the load_package function from the install.load
# package (it is assumed that you have already installed these packages)

import::from(fpCompare, "%==%")
```

Fraction (or Mixed number) to a Decimal (Numeric Vector)

Example 1

Convert the following mixed number values to a decimal number. The non-numeric (character) values are ignored with the frac_to_numeric function.

```
trxt <- "1 1/3"

frac_to_numeric(trxt)

## [1] 1.333333
tlrxy <- "4 1/8 inches"

frac_to_numeric(tlrxy)

## [1] 4.125
tmrxy <- "12 13/16 inches"

frac_to_numeric(tmrxy)

## [1] 12.8125
hjtevo <- "28/3 inches"

frac_to_numeric(hjtevo)

## [1] 9.333333
```

Construction Decimal (Measurement in feet inch fraction returned as a decimal value)

Example 1 (Convert a feet inch fraction measurement to its decimal equivalent)

In the following example, the expressed values are the same in t1 through t16 (34 feet 3 1/2 inches); however, the only difference exists in the character strings. The point here is to show that the character values do not really matter because it is the numeric values that actually determine the decimal equivalent. The decimal equivalent will be expressed in decimal feet when the result is traditional and it will be expressed in decimal inches when the result is librecad.

```
t1 <- "34'-3 1/2\""
t2 <- "34-3 1/2\""
t3 <- "34' 3 1/2\""
t4 <- "34'-3 1/2"
t5 <- "34-3 1/2"
t6 <- "34 3 1/2"
t7 <- "34 ft 3 1/2 in"
t8 <- "34 3 1/2"
t9 <- "34 fts 3 1/2 in"
t10 <- "34 foot 3 1/2 in"
t11 <- "34 foot 3 1/2 inch"
t12 <- "34 foot 3 1/2 in"
t13 <- "34 feet 3 1/2 in"
t14 <- "34 feet 3 1/2 inch"
t15 <- "34 feet 3 1/2 in"
t16 <- "34 FEEt 3 1/2 IN"
```

a) Multiply t1 by t2 and return the answer in acres

```
(construction_decimal(t1, result = "traditional", output = "vector") * construction_decimal(t2,
    result = "traditional", output = "vector") * 4)/43560
```

```
## [1] 0.1079815
# acres
```

b) Square t1 and return the answer in acres

```
(construction_decimal(t1, result = "traditional", output = "vector")^2 * 4)/43560
```

```
## [1] 0.1079815
# acres
```

c) Both a) and b) can be expressed using the following as well

```
t1_ft2 <- set_units((construction_decimal(t1, result = "traditional", output = "vector") *
    construction_decimal(t2, result = "traditional", output = "vector") * 4), US_survey_foot^2)

t1_acres <- t1_ft2

units(t1_acres) <- make_units(acre)
t1_acres
```

```
## 0.1079815 [acre]
```

```
t1_ft2s <- set_units((construction_decimal(t1, result = "traditional", output = "vector")^2 *
    4), US_survey_foot^2)

t1_acress <- t1_ft2s

units(t1_acress) <- make_units(acre)
t1_acress
```

```
## 0.1079815 [acre]
```

d) Return the sum of t1 - t5

```
sum(construction_decimal(t1, result = "traditional", output = "vector"), construction_decimal(t2,
    result = "traditional", output = "vector"), construction_decimal(t3, result = "traditional",
    output = "vector"), construction_decimal(t4, result = "traditional", output = "vector"),
    construction_decimal(t5, result = "traditional", output = "vector"))
```

```
## [1] 171.4583
```

e) Return the traditional vector result for t1 - t16

```
construction_decimal(t1, result = "traditional", output = "vector")
```

```
## [1] 34.29167
```

```
construction_decimal(t2, result = "traditional", output = "vector")
```

```
## [1] 34.29167
```

```
construction_decimal(t3, result = "traditional", output = "vector")
```

```
## [1] 34.29167
```

```
construction_decimal(t4, result = "traditional", output = "vector")
```

```
## [1] 34.29167
```

```
construction_decimal(t5, result = "traditional", output = "vector")
```

```
## [1] 34.29167
```

```
construction_decimal(t6, result = "traditional", output = "vector")
```

```
## [1] 34.29167
```

```
construction_decimal(t7, result = "traditional", output = "vector")
```

```
## [1] 34.29167
```

```
construction_decimal(t8, result = "traditional", output = "vector")
```

```
## [1] 34.29167
```

```
construction_decimal(t9, result = "traditional", output = "vector")
```

```
## [1] 34.29167
```

```
construction_decimal(t10, result = "traditional", output = "vector")
```

```
## [1] 34.29167
```

```
construction_decimal(t11, result = "traditional", output = "vector")
```

```
## [1] 34.29167
```

```
construction_decimal(t12, result = "traditional", output = "vector")
```

```
## [1] 34.29167
```

```
construction_decimal(t13, result = "traditional", output = "vector")
```

```
## [1] 34.29167
```

```
construction_decimal(t14, result = "traditional", output = "vector")
```

```
## [1] 34.29167
```

```
construction_decimal(t15, result = "traditional", output = "vector")
```

```
## [1] 34.29167
```

```
construction_decimal(t16, result = "traditional", output = "vector")
```

```
## [1] 34.29167
```

f) Return the librecad vector result result for t1 - t16

```
construction_decimal(t1, result = "librecad", output = "vector")
```

```
## [1] 411.5
```

```
construction_decimal(t2, result = "librecad", output = "vector")
```

```
## [1] 411.5
```

```
construction_decimal(t3, result = "librecad", output = "vector")
```

```
## [1] 411.5
```

```
construction_decimal(t4, result = "librecad", output = "vector")
```

```
## [1] 411.5
```

```
construction_decimal(t5, result = "librecad", output = "vector")
```

```
## [1] 411.5
```

```
construction_decimal(t6, result = "librecad", output = "vector")
```

```
## [1] 411.5
```

```
construction_decimal(t7, result = "librecad", output = "vector")
```

```
## [1] 411.5
```

```
construction_decimal(t8, result = "librecad", output = "vector")
```

```
## [1] 411.5
```

```
construction_decimal(t9, result = "librecad", output = "vector")
```

```
## [1] 411.5
```

```
construction_decimal(t10, result = "librecad", output = "vector")
```

```
## [1] 411.5
```

```
construction_decimal(t11, result = "librecad", output = "vector")
```

```
## [1] 411.5
```

```
construction_decimal(t12, result = "librecad", output = "vector")
```

```
## [1] 411.5
```

```
construction_decimal(t13, result = "librecad", output = "vector")
```

```
## [1] 411.5
```

```
construction_decimal(t14, result = "librecad", output = "vector")
```

```
## [1] 411.5
```

```
construction_decimal(t15, result = "librecad", output = "vector")
```

```
## [1] 411.5
```

```
construction_decimal(t16, result = "librecad", output = "vector")
```

```
## [1] 411.5
```

Example 2 [Verify the sum]

In the following example, verify that the sum of m1 through m6 [33 feet 3 1/2 inches, 32 feet 1 inch, 32 feet 1 inch, 32 feet 1 inch, 32 feet 1 inch, 33 feet 3 1/2 inches] using the construction_decimal function is equivalent to 194 feet 11 inches. As long as the last value is TRUE, then the verification is complete.

```
m1 <- "33'-3 1/2\""
m2 <- "32'-1"
m3 <- "32'-1"
m4 <- "32'-1"
m5 <- "32'-1"
```

```
m6 <- "33'-3 1/2\""

msum <- sum(construction_decimal(m1, result = "traditional", output = "vector"),
    construction_decimal(m2, result = "traditional", output = "vector"), construction_decimal(m3,
        result = "traditional", output = "vector"), construction_decimal(m4, result = "traditional",
        output = "vector"), construction_decimal(m5, result = "traditional", output = "vector"),
    construction_decimal(m6, result = "traditional", output = "vector"))

# print msum as a decimal number
msum
```

```
## [1] 194.9167
```

```
# print the construction fraction for msum
construction_fraction(msum, type = "traditional", result = "traditional", fraction = 0)
```

```
## [1] "194 ft 11 in"
```

```
# check whether msum is equal to the decimal expressed by 194 feet 11 inches or
# not
construction_decimal("194'-11", result = "traditional", output = "vector") %==% msum
```

```
## [1] TRUE
```

Example 3 [Perimeter sum]

Calculate the sum of an object with the following perimeter measurements [3 inches, 8 inches, 6 inches, 2 5/8 inches, 2 feet 6 3/4 inches, 2 feet 6 3/4 inches, 2 feet 6 3/4 inches, 2 5/8 inches]. The sum will be in decimal feet.

```
sum(construction_decimal("0 3", result = "traditional", output = "vector"), construction_decimal("0 8",
    result = "traditional", output = "vector"), construction_decimal("0 6", result = "traditional",
    output = "vector")) * sum(construction_decimal("0 2 5/8", result = "traditional",
    output = "vector"), 3 * construction_decimal("2 6 3/4", result = "traditional",
    output = "vector"), construction_decimal("0 2 5/8", result = "traditional", output = "vector"))
```

```
## [1] 11.51042
```

Example 4 [Total surface area & volume for riprap]

Calculate the amount of surface area fill, in acres, for riprap placement and the volume of fill in yards^3 along a streambank.

```
bank <- set_units(construction_decimal("72 3 1/3", result = "traditional", output = "vector"),
    US_survey_foot)
# 72 feet 3 1/3 inches

bank
```

```
## 72.27778 [US_survey_foot]
```

```
riprap <- set_units(construction_decimal("0 15.0", result = "traditional", output = "vector"),
    US_survey_foot)

riprap

## 1.25 [US_survey_foot]
riprap_yd <- riprap

units(riprap_yd) <- make_units(yd)

riprap_yd

## 0.4166675 [yd]
OHWM_width <- set_units(25, US_survey_foot)

OHWM_width

## 25 [US_survey_foot]
width <- set_units(47, US_survey_foot)

width

## 47 [US_survey_foot]
bank_area1 <- width * bank

bank_area1

## 3397.056 [US_survey_foot^2]
bank_area2 <- bank_area1

units(bank_area2) <- make_units(yd^2)

bank_area2

## 377.4521 [yd^2]
bank_area3 <- bank_area1

units(bank_area3) <- make_units(acres)

bank_area3

## 0.07798566 [acres]
vol_bank <- riprap_yd * bank_area2

vol_bank

## 157.272 [yd^3]
bank_area_OHWM1 <- OHWM_width * bank

bank_area_OHWM1

## 1806.944 [US_survey_foot^2]
```

```
bank_area_OHWM2 <- bank_area1

units(bank_area_OHWM2) <- make_units(yd^2)

bank_area_OHWM2

## 377.4521 [yd^2]
bank_area_OHWM3 <- bank_area1

units(bank_area_OHWM3) <- make_units(acres)

bank_area_OHWM3

## 0.07798566 [acres]
vol_bank_OHWM <- riprap_yd * bank_area_OHWM2

vol_bank_OHWM

## 157.272 [yd^3]
fill_ft2 <- bank_area_OHWM1

fill_ft2

## 1806.944 [US_survey_foot^2]
fill_acres <- bank_area_OHWM3

fill_acres

## 0.07798566 [acres]
fill_yd2 <- bank_area_OHWM2

fill_yd2

## 377.4521 [yd^2]
fill_yd3 <- vol_bank_OHWM

fill_yd3

## 157.272 [yd^3]
```

Example 5 [Examples from the Spike reference]

```
psst <- "7' 4 5/16\""

pssts <- "0 3 3/8\""

wall1 <- "12' 7\""

wall2 <- "40' 9\""
```

```
construction_decimal(psst, result = "traditional", output = "vector")
```

```
## [1] 7.359375
```

```
construction_decimal(pssts, result = "traditional", output = "vector")
```

```
## [1] 0.28125
```

```
construction_decimal(wall1, result = "traditional", output = "vector")
```

```
## [1] 12.58333
```

```
construction_decimal(wall2, result = "traditional", output = "vector")
```

```
## [1] 40.75
```

a) Matching the decimal value for 3 3/8 inches

Since pssts is a fraction representing 3 3/8 inches, it is better to use the frac_to_numeric function or the construction_decimal function with the librecad result instead of using the construction_decimal function with the traditional result to match the decimal value from the Spike reference.

```
pssts1 <- "3 3/8\""
```

```
frac_to_numeric(pssts1)
```

```
## [1] 3.375
```

```
# or more simply
```

```
pssts1b <- "3 3/8 in"
```

```
# checks
frac_to_numeric(pssts1b)
```

```
## [1] 3.375
```

```
frac_to_numeric(pssts1) %==% frac_to_numeric(pssts1b)
```

```
## [1] TRUE
```

```
frac_to_numeric(pssts1) %==% construction_decimal(pssts, result = "librecad", output = "vector")
```

```
## [1] TRUE
```

```
frac_to_numeric(pssts1b) %==% construction_decimal(pssts, result = "librecad", output = "vector")
```

```
## [1] TRUE
```

Example 6 [Print a table with equivalent length measurements]

In a tabular format, show the equivalents to 1 feet 2 7/16 inches, 6 feet 8 3/4 inches, 6 feet 5 3/256 inches in decimal inches, decimal feet, decimal yards, decimal millimeters, decimal centimeters, and decimal meters.

```
librecad1 <- "1 2 7/16\""

construction_decimal(librecad1, result = "traditional", output = "vector")
```

```
## [1] 1.203125
```

```
knitr::kable(format(construction_decimal(librecad1, result = "traditional", output = "table"),
    digits = 6, nsmall = 0))
```

Measurement	Units
14.437500	in
1.203125	ft
0.401042	yd
366.712500	mm
36.671250	cm
0.366712	m

```
construction_decimal(librecad1, result = "librecad", output = "vector")
```

```
## [1] 14.4375
```

```
knitr::kable(format(construction_decimal(librecad1, result = "librecad", output = "table"),
    digits = 4, nsmall = 0))
```

Measurement	Units
14.4375	in
1.2031	ft
0.4010	yd
366.7125	mm
36.6713	cm
0.3667	m

```
librecad2 <- "6' 8 3/4 in"

construction_decimal(librecad2, result = "traditional", output = "vector")
```

```
## [1] 6.729167
```

```
knitr::kable(format(construction_decimal(librecad2, result = "traditional", output = "table"),
    digits = 6, nsmall = 6))
```

Measurement	Units
80.750000	in
6.729167	ft
2.243056	yd
2051.050000	mm
205.105000	cm
2.051050	m

```
construction_decimal(librecad2, result = "librecad", output = "vector")
```

```
## [1] 80.75
```

```
knitr::kable(format(construction_decimal(librecad2, result = "librecad", output = "table"),
    digits = 2, nsmall = 2))
```

Measurement	Units
80.75	in
6.73	ft
2.24	yd
2051.05	mm
205.10	cm
2.05	m

```
librecad3 <- "6'-5 3/256\""
```

```
construction_decimal(librecad3, result = "traditional", output = "vector")
```

```
## [1] 6.417643
```

```
knitr::kable(format(construction_decimal(librecad3, result = "traditional", output = "table"),
    digits = 6, nsmall = 6))
```

Measurement	Units
77.011719	in
6.417643	ft
2.139214	yd
1956.097656	mm
195.609766	cm
1.956098	m

```
construction_decimal(librecad3, result = "librecad", output = "vector")
```

```
## [1] 77.01172
```

```
knitr::kable(format(construction_decimal(librecad3, result = "librecad", output = "table"),
    digits = 5, nsmall = 5))
```

Measurement	Units
77.01172	in
6.41764	ft
2.13921	yd
1956.09766	mm
195.60977	cm
1.95610	m

Construction Fraction

Example 1

Return the fractional equivalents for the decimal value of 6 feet 5 3/256 inches.

```
checker <- "6'-5 3/256 in"
```

```
checkers <- construction_decimal(checker, result = "traditional", output = "vector")
```

```
checkers
```

```
## [1] 6.417643
```

```
construction_fraction(checkers, type = "traditional", result = "traditional", fraction = 0)
```

```
## [1] "6 ft 5 in"
```

```
construction_fraction(checkers, type = "traditional", result = "traditional", fraction = 2)
```

```
## [1] "6 ft 5 0/2 in"
```

```
construction_fraction(checkers, type = "traditional", result = "traditional", fraction = 4)
```

```
## [1] "6 ft 5 0/4 in"
```

```
construction_fraction(checkers, type = "traditional", result = "traditional", fraction = 8)
```

```
## [1] "6 ft 5 0/8 in"
```

```
construction_fraction(checkers, type = "traditional", result = "traditional", fraction = 16)
```

```
## [1] "6 ft 5 0/16 in"
```

```
construction_fraction(checkers, type = "traditional", result = "traditional", fraction = 32)
```

```
## [1] "6 ft 5 0/32 in"
```

```
construction_fraction(checkers, type = "traditional", result = "traditional", fraction = 64)
```

```
## [1] "6 ft 5 1/64 in"
```

```
construction_fraction(checkers, type = "traditional", result = "traditional", fraction = 100)
```

```
## [1] "6 ft 5 1/100 in"
```

```
construction_fraction(checkers, type = "traditional", result = "traditional", fraction = 128)
```

```
## [1] "6 ft 5 2/128 in"
```

```
construction_fraction(checkers, type = "traditional", result = "traditional", fraction = 256)
```

```
## [1] "6 ft 5 3/256 in"
```

Example 2

Return the fractional equivalents for the decimal value of 77.6875 inches.

```
checkin <- 77.6875

construction_fraction(checkin, type = "librecad", result = "traditional", fraction = 0)

## [1] "6 ft 5 in"
construction_fraction(checkin, type = "librecad", result = "traditional", fraction = 2)

## [1] "6 ft 5 1/2 in"
construction_fraction(checkin, type = "librecad", result = "traditional", fraction = 4)

## [1] "6 ft 5 3/4 in"
construction_fraction(checkin, type = "librecad", result = "traditional", fraction = 8)

## [1] "6 ft 5 5/8 in"
construction_fraction(checkin, type = "librecad", result = "traditional", fraction = 16)

## [1] "6 ft 5 11/16 in"
construction_fraction(checkin, type = "librecad", result = "traditional", fraction = 32)

## [1] "6 ft 5 22/32 in"
construction_fraction(checkin, type = "librecad", result = "traditional", fraction = 64)

## [1] "6 ft 5 44/64 in"
construction_fraction(checkin, type = "librecad", result = "traditional", fraction = 100)

## [1] "6 ft 5 69/100 in"
construction_fraction(checkin, type = "librecad", result = "traditional", fraction = 128)

## [1] "6 ft 5 88/128 in"
construction_fraction(checkin, type = "librecad", result = "traditional", fraction = 256)

## [1] "6 ft 5 176/256 in"
```

Example 3

Given the lengths of 5 feet 1 3/4 inches and 21 feet 7 3/8. Multiply, divide, add, and subtract the value to compare the results to the CalculatorSoup.com reference.

a) Multiplication

The answer is "111.225 ft^2 (16016.4063 in^2)".

```
length1 <- "5 feet 1 3/4 inches"
length2 <- "21 feet 7 3/8"

length_product_in <- construction_decimal(length1, result = "librecad", output = "vector") *
    construction_decimal(length2, result = "librecad", output = "vector")
```

```
length_product_in
```

```
## [1] 16016.41
```

```
length_product_ft <- construction_decimal(length1, result = "traditional", output = "vector") *
    construction_decimal(length2, result = "traditional", output = "vector")

length_product_ft
```

```
## [1] 111.225
```

```
construction_fraction(length_product_ft, type = "traditional", result = "traditional",
    fraction = 8)
```

```
## [1] "111 ft 2 6/8 in"
```

```
# Alternatively, this could all be done in a single step as well:

construction_fraction((construction_decimal(length1, result = "traditional", output = "vector") *
    construction_decimal(length2, result = "traditional", output = "vector")), type = "traditional",
    result = "traditional", fraction = 8)
```

```
## [1] "111 ft 2 6/8 in"
```

b) Division

The answer is "0.2381".

```
length1 <- "5 feet 1 3/4 inches"
length2 <- "21 feet 7 3/8"

length_quotient <- construction_decimal(length1, result = "librecad", output = "vector")/construction_d
    result = "librecad", output = "vector")

length_quotient
```

```
## [1] 0.2380723
```

```
construction_fraction(length_quotient, type = "traditional", result = "traditional",
    fraction = 8)
```

```
## [1] "0 ft 2 7/8 in"
```

```
# Alternatively, this could all be done in a single step as well:

construction_fraction((construction_decimal(length1, result = "traditional", output = "vector")/construc
    result = "traditional", output = "vector")), type = "traditional", result = "traditional",
    fraction = 8)
```

```
## [1] "0 ft 2 7/8 in"
```

c) Addition

The answer is "26 ft 9 1/8 in".

```
length_sum <- sum(construction_decimal(length1, result = "traditional", output = "vector"),
    construction_decimal(length2, result = "traditional", output = "vector"))

construction_fraction(length_sum, type = "traditional", result = "traditional", fraction = 8)
```

```
## [1] "26 ft 9 1/8 in"
```

```
# Alternatively, this could all be done in a single step as well:

construction_fraction(sum(construction_decimal(length1, result = "traditional", output = "vector"),
    construction_decimal(length2, result = "traditional", output = "vector")), type = "traditional",
    result = "traditional", fraction = 8)
```

```
## [1] "26 ft 9 1/8 in"
```

d) Subtraction (Negative result)

The answer is "-16 ft 5 5/8 in".

```
length_difference1 <- construction_decimal(length1, result = "traditional", output = "vector") -
    construction_decimal(length2, result = "traditional", output = "vector")

construction_fraction(length_difference1, type = "traditional", result = "traditional",
    fraction = 8)
```

```
## [1] "-16 ft -5 -5/8 in"
```

```
# Alternatively, this could all be done in a single step as well:

construction_fraction((construction_decimal(length1, result = "traditional", output = "vector") -
    construction_decimal(length2, result = "traditional", output = "vector")), type = "traditional",
    result = "traditional", fraction = 8)
```

```
## [1] "-16 ft -5 -5/8 in"
```

d) Subtraction (Positive result)

```
length_difference2 <- construction_decimal(length2, result = "traditional", output = "vector") -
    construction_decimal(length1, result = "traditional", output = "vector")

construction_fraction(length_difference2, type = "traditional", result = "traditional",
    fraction = 8)
```

```
## [1] "16 ft 5 5/8 in"
```

```
# Alternatively, this could all be done in a single step as well:

construction_fraction((construction_decimal(length2, result = "traditional", output = "vector") -
    construction_decimal(length1, result = "traditional", output = "vector")), type = "traditional",
    result = "traditional", fraction = 8)
```

```
## [1] "16 ft 5 5/8 in"
```

Example 4

"Imagine you've bought a wooden panel, 5 meters long. You plan to cut it into six equal parts, but you have only a tape measure with the fractional inches scale. We can measure the length with precision down to 1/32"." From the Omni Calculator: Inches to Fraction Calculator reference.

What is the length for each of the 6 boards cut?

```
panel <- set_units(5, "m")

panel
```

```
## 5 [m]
```

```
panel_ft <- panel

units(panel_ft) <- make_units(ft)

panel_ft
```

```
## 16.4042 [ft]
```

```
panel_6 <- panel_ft/6

panel_6
```

```
## 2.734033 [ft]
```

```
construction_fraction(drop_units(panel_6), type = "traditional", result = "traditional",
    fraction = 16)
```

```
## [1] "2 ft 8 13/16 in"
```

```
construction_fraction(drop_units(panel_6), type = "traditional", result = "traditional",
    fraction = 32)
```

```
## [1] "2 ft 8 26/32 in"
```

Construction Decimal Engineering (LibreCAD Style)

Examples

```
librecad1a <- "6' 8 3/4 in"

construction_decimal_eng(librecad1a)
```

```
## [1] "6'-8.75\""
```

```
librecad2a <- "6'-5 3/256\""

construction_decimal_eng(librecad2a)
```

```
## [1] "6'-5.01171875\""
```

R Help for iemisc Functions

Please refer to the `iemisc` [https://CRAN.R-project.org/package=iemisc] help definitions for the `frac_to_numeric`, `construction_decimal`, and `construction_fraction` functions below:

```
## Registered S3 method overwritten by 'printr':
##   method                from
##   knit_print.data.frame rmarkdown
```

```
## <environment: namespace:printr>
```

```
help(frac_to_numeric, package = "iemisc")
```

Fraction (or Mixed number) to a Decimal (Numeric Vector)

Description:

```
 Converts a fraction or a mixed number to a decimal
```

Usage:

```
 frac_to_numeric(n)
```

Arguments:

```
   n: character vector that contains the fraction or mixed number
      (can also include text, ex. inch, inches, etc. that will be
      removed from the vector)
```

Value:

```
 the numeric 'vector' as a decimal
```

Note:

```
 If you have a measurement in feet + inches, then use
 'construction_fraction' instead.
```

Author(s):

```
 Irucka Embry
```

Source:

```
 removing all non-numeric characters from a string, but not "." - R
 help on nabble.com answered by David Winsemius on Jul 26, 2016.
 See
 <https://web.archive.org/web/20190730141421/http://r.789695.n4.nabble.com/removing-all-non-numeric-cha
 Retrieved thanks to the Internet Archive: Wayback Machine.
```

References:

```
   1. Bill Venables, 2016-02-10, "Vulgar Fractions in R",
      fractional vignette,
      <https://CRAN.R-project.org/package=fractional/vignettes/Vulgar_Fractions_in_R.html>.

   2. The Home Depot, 9 December 2022, "How to Read a Tape
      Measure", <https://archive.vn/fhBmg>. Provided the
      archive.today webpage capture for The Home Depot URL for
      acceptance into CRAN.
```

3. Wikimedia Foundation, Inc. Wikipedia, 29 December 2021, "Pi",
<https://en.wikipedia.org/wiki/Pi>.

Examples:

```
# Please refer to the iemisc: Construction Measurement Examples vignette for
# additional examples

# Example 1 -- Reference 1

library(iemisc)

xx <- as.character(fractional::fractional(1:9 / 12))

try(frac_to_numeric(xx))
# Please note that there will be an error because this function is designed to
# only process one fraction at a time.

lapply(xx, frac_to_numeric)
# Please note that this is the correct way to work with several fractions at once.




# Example 2

library(iemisc)

xi <- fracture::fracture((50:65) / 12)

try(frac_to_numeric(xi))
# Please note that there will be an error because this function is designed to
# only process one fraction at a time.

lapply(xi, frac_to_numeric)
# Please note that this is the correct way to work with several fractions at once.




# Example 3

library(iemisc)

xyy <- fracture::fracture((1:11) / 12)

try(frac_to_numeric(xyy))
# Please note that there will be an error because this function is designed to
# only process one fraction at a time.

lapply(xyy, frac_to_numeric)
# Please note that this is the correct way to work with several fractions at once.
```

```
# Example 4

library(iemisc)

xft <- as.character(MASS::fractions((1:70) / 12))

try(frac_to_numeric(xft))
# Please note that there will be an error because this function is designed to
# only process one fraction at a time.

lapply(xft, frac_to_numeric)
# Please note that this is the correct way to work with several fractions at once.




# Example 5

library(iemisc)

pix <- "270/11"

pi1 <- "22/7" # Reference 3

pi2 <- "355/113" # Reference 3

frac_to_numeric(pix)

frac_to_numeric(pi1)

frac_to_numeric(pi2)




# Example 6

# If you have a construction measurement that includes a dimension in feet,
# such as 49 ft 7 5/8 in, don't use the frac_to_numeric function, instead
# use the construction_fraction function.

library(iemisc)

xxift <- "49 ft 7 5/8 in"

construction_decimal(xxift, result = "traditional", output = "vector")




# Example 7 -- Reference 2
```

```
truss_marks <- "19 3/16 inches"

frac_to_numeric(truss_marks)
```

```
help(construction_decimal, package = "iemisc")
```

Construction Decimal

Description:

```
Convert a construction measurement in US Customary Units (foot +
inch) with or without a fraction into its equivalent as a decimal
```

Usage:

```
construction_decimal(
  measurement,
  result = c("traditional", "librecad"),
  output = c("vector", "table")
)
```

Arguments:

measurement: character vector that contains the construction measurement (foot + inch)

result: character vector that contains the decimal type options are traditional (ex. 1.203125 = 1'-2 7/16" where the whole number is the value in ft and the decimal is the value in inches & librecad (ex. 14.43112 = 1'-2 7/16"), whereby LibreCAD defines its decimal unit as "integer part separated from the fractional part of a number by a decimal". Thus, both the whole number and the decimal is the value in inches.

output: character vector that contains the type of output. The options are vector (just the single value as a decimal) and table the decimal value in inch (in), feet (ft), yard (yd), millimeters (mm), centimeters (cm), and meters (m).

Value:

```
the construction measurement value as a numeric 'vector' as a
decimal or as a table (depends on the output parameters)
```

Note:

```
If you only have a measurement in inches, then use
'frac_to_numeric' instead.
```

Author(s):

```
Irucka Embry
```

Source:

1. removing all non-numeric characters from a string, but not "." - R help on nabble.com answered by David Winsemius on Jul 26, 2016. See <https://web.archive.org/web/20190730141421/http://r.789695.n4.nabble.com/removing-all-non-numeri
   Retrieved thanks to the Internet Archive: Wayback Machine

2. r - How to not run an example using roxygen2? - Stack Overflow answered and edited by samkart on Jul 9 2017. (Also see the additional comments in response to the answer.) See <https://stackoverflow.com/questions/12038160/how-to-not-run-an-example-using-roxygen2>.

3. devtools - Issues in R package after CRAN asked to replace

dontrun by donttest - Stack Overflow answered by Hong Ooi on Sep 1 2020. (Also see the additional comments in response to the answer.) See <https://stackoverflow.com/questions/63693563/issues-in-r-package-after-cran-asked-to-replace-don

4. regex - Replace single backslash in R - Stack Overflow answered and edited by Hong Ooi on Aug 21, 2014. (Also see the additional comments in response to the answer.) See <https://stackoverflow.com/questions/25424382/replace-single-backslash-in-r>.

References:

1. LibreCAD v2.2.0 - User Manual - Fundamentals: Units, 7 May 2022, <https://librecad-docs-dev.readthedocs.io/en/latest/ref/fundamentals.html#units>.

2. Spike, 1 January 2022, "Foot and Inch to Decimal Format Conversion", <https://www.spikevm.com/calculators/fraction-decimal-calculators.php>.

Examples:

```
# Please refer to the iemisc: Construction Measurement Examples vignette for
# additional examples

# Example 1

library(iemisc)

construction_decimal("2'-0\"", result = "traditional", output = "vector")

construction_decimal("1'-2 7/16\"", result = "librecad", output = "vector")



# Example 2

library(iemisc)

construction_decimal("0 6", result = "traditional", output = "vector")

construction_decimal("0 6", result = "librecad", output = "vector")



# Example 3

library(iemisc)

tss1 <- "48'-0 1/2\""
tss2 <- "56-9 1/2\""

sum(construction_decimal(tss1, result = "traditional", output = "vector"),
construction_decimal(tss2, result = "traditional", output = "vector"))
```

```
# See Source 2 and Source 3

# Example 4

library(iemisc)

try(construction_decimal(5, result = "traditional", output =
"vector")) # please see the error message

ex_error <- character(0)
try(construction_decimal(ex_error, result = "traditional",
output = "vector")) # please see the error message

try(construction_decimal(NA, result = "traditional", output =
"vector")) # please see the error message

try(construction_decimal("feet", result = "traditional", output =
"vector")) # please see the error message




# Example 5

library(iemisc)

app1 <- "5' 2\""

app2 <- "6' 3\""

app3 <- construction_decimal(app1, result = "traditional", output = "vector") *
construction_decimal(app2, result = "traditional", output = "vector")
app3

# If you want to have the fractional value using 16ths, do the following:

construction_fraction(app3, type = "traditional", result = "traditional",
fraction = 16)
help(construction_fraction, package = "iemisc")
```

Construction Fraction

Description:

```
Convert a construction measurement in US Customary Units as a
decimal into its nearest equivalent as foot + inch with or without
a fraction
```

Usage:

```
construction_fraction(
  measurement,
  type = c("traditional", "librecad"),
  result = c("traditional", "inch"),
  fraction = c(0, 2, 4, 8, 16, 32, 64, 100, 128, 256)
```

)

Arguments:

measurement: numeric vector that contains the construction measurement as a decimal

type: character vector that contains the decimal type options are traditional (ex. 1.203125 = 1'-2 7/16\" where the whole number is the value in ft and the decimal is the value in inches & librecad (ex. 14.4375 = 1'-2 7/16\"), whereby LibreCAD defines its decimal unit as "integer part separated from the fractional part of a number by a decimal". Thus, both the whole number and the decimal is the value in inches.

result: character vector that contains the resulting fraction type (options are traditional (ex. 1.203125 = 1 ft 2 7/16 in) & inch (ex. 14.4375 = 14 7/16 in)). This is the same as fractional (fractional inch) in LibreCAD.

fraction: numeric vector that contains the fractional part to return. The options are 0, 2, 4, 8, 16, 32, 64, 100, 128, or 256.

Value:

the construction measurement value as a character 'vector' as foot + fraction of an inch or as inch + fraction of an inch (depends on the parameters)

Author(s):

Irucka Embry

References:

1. Spike, 7 May 2022, "How to Convert Feet in Decimal Format to Foot, Inch and Fraction Values", <https://www.spikevm.com/construction-math/convert-decimal-fraction.php>.

2. myCarpentry, 7 May 2022, "Online Fraction Calculator", <https://www.mycarpentry.com/online-fraction-calculator.html>.

3. LibreCAD, User Manual - Fundamentals: Units: Architectural and Decimal, 7 May 2022, <https://librecad-docs-dev.readthedocs.io/en/latest/ref/fundamentals.html#units>.

4. Inch Calculator. Inch Fraction Calculator - Convert Decimal to Inches, 9 May 2022, <https://www.inchcalculator.com/inch-fraction-calculator/>.

Examples:

```
# Please refer to the iemisc: Construction Measurement Examples vignette for
# additional examples

library(iemisc)

# Example 1 from the Spike Reference

check1 <- 18.649 # decimal feet

construction_fraction(check1, type = "traditional", result =
```

```
"traditional", fraction = 16)

# Reverse the calculation to check out the absolute error

check2 <- construction_decimal(construction_fraction(check1,
type = "traditional", result = "traditional", fraction = 16),
result = "traditional", output = "vector")

fracture::fracture(check2 - check1) # difference in inches

# by approximate error

approxerror(check2, check1) # answer as a percent (\%)


# check all other fraction levels

construction_fraction(check1, type = "traditional", result =
"traditional", fraction = 0)

construction_fraction(check1, type = "traditional", result =
"traditional", fraction = 2)

construction_fraction(check1, type = "traditional", result =
"traditional", fraction = 4)

construction_fraction(check1, type = "traditional", result =
"traditional", fraction = 8)

construction_fraction(check1, type = "traditional", result =
"traditional", fraction = 32)

construction_fraction(check1, type = "traditional", result =
"traditional", fraction = 64)

construction_fraction(check1, type = "traditional", result =
"traditional", fraction = 100)

construction_fraction(check1, type = "traditional", result =
"traditional", fraction = 128)




# Example 2

library(iemisc)
import::from(fpCompare, "%==%")


x1 <- construction_fraction(1.203125, type = "traditional", result =
"traditional", fraction = 16)
```

```
x2 <- construction_fraction(14.4375, type = "librecad", result =
"inch", fraction = 16)

x3 <- construction_fraction(14.4375, type = "librecad", result =
"traditional", fraction = 16)

x4 <- construction_fraction(14.43112, type = "librecad", result =
"traditional", fraction = 16)

x5 <- construction_fraction(14.43112, type = "librecad", result =
"inch", fraction = 16)


ex1 <- frac_to_numeric(x2)

ex2 <- construction_decimal(x1, result = "librecad", output = "vector")

ex3 <- construction_decimal(x3, result = "librecad", output = "vector")

ex4 <- construction_decimal(x4, result = "librecad", output = "vector")

ex5 <- frac_to_numeric(x5)


# check if ex1, ex2, ex3, ex4, and ex5 are equivalent
ex1 %==% ex2

ex1 %==% ex3

ex1 %==% ex4

ex1 %==% ex5

ex2 %==% ex3

ex2 %==% ex4

ex2 %==% ex5

ex3 %==% ex4

ex3 %==% ex5

ex4 %==% ex5




# Example 3 (from the Inch Calculator Reference)

library(iemisc)

construction_fraction(2.695, type = "librecad", result = "traditional",
fraction = 16)
```

```
construction_fraction(2.695, type = "librecad", result = "inch",
fraction = 16)
```

```
# Example 4

library(iemisc)

construction_fraction(17.71354, type = "traditional", result = "traditional",
fraction = 16)

construction_fraction(17.71354, type = "traditional", result = "inch",
fraction = 16)
```

Works Cited

Furey, Edward "Feet and Inches Calculator" at https://www.calculatorsoup.com/calculators/construction/feetandinches.php from CalculatorSoup, https://www.calculatorsoup.com - Online Calculators, Last updated: November 12, 2018

Omni Calculator, "Inches to Fraction Calculator", Created by Wojciech Sas, PhD, Last updated: Jun 05, 2023, https://www.omnicalculator.com/conversion/inches-to-fraction.

Spike, 1 January 2022, "Foot and Inch to Decimal Format Conversion", https://www.spikevm.com/calculators/fraction-decimal-calculators.php.

EcoC²S Links

EcoC²S Home – https://www.ecoccs.com/
About EcoC²S – https://www.ecoccs.com/about_ecoc2s.html
Services – https://www.ecoccs.com/services.html
1 Stop Shop – https://www.ecoccs.com/other_biz.html
Products – https://www.questionuniverse.com/products.html
Media – https://www.ecoccs.com/media.html
Resources – https://www.ecoccs.com/resources.html
R Trainings and Resources provided by EcoC²S (Irucka Embry, E.I.T.) – https://www.ecoccs.com/rtraining.html

Copyright and License