

# Package: birk (via r-universe)

August 23, 2024

**Type** Package

**Title** MA Birk's Functions

**Version** 2.1.2

**Date** 2016-07-27

**Author** Matthew A. Birk

**Maintainer** Matthew A. Birk <matthewabirk@gmail.com>

**Description** Collection of tools to make R more convenient. Includes tools to summarize data using statistics not available with base R and manipulate objects for analyses.

**Imports** grDevices, stats

**License** GPL-3

**Encoding** UTF-8

**RoxygenNote** 5.0.1

**NeedsCompilation** no

**Date/Publication** 2016-07-27 07:56:39

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

**RemoteUrl** <https://github.com/cran/birk>

**RemoteRef** HEAD

**RemoteSha** 24e8c23637fe365578b20c6854d85af788cc96a2

## Contents

birk	2
conv_dim	2
conv_unit	3
conv_unit_options	5
geom_mean	6
range_seq	7
se	8
summ_stat	8
which.closest	9

**Index****11**


---

birk	<i>MA Birk's Functions</i>
------	----------------------------

---

**Description**

Collection of tools to make R more convenient. Includes tools to summarize data using statistics not available with base R and manipulate objects for analyses.

**Author(s)**

Matthew A. Birk, <matthewabirk@gmail.com>

---

conv_dim	<i>Convert Dimensions of Measurement</i>
----------	--

---

**Description**

DEPRECATED. Converts between dimensions of measurement given a transition dimension (the dimension that "bridges" x and y, e.g. liters per second, lbs per acre). Note that 2 of the 3 measurements (x, y, or trans) must be defined to calculate the 3rd. See [conv\\_unit\\_options](#) for all options.

**Usage**

```
conv_dim(x, x_unit, trans, trans_unit, y, y_unit)
```

**Arguments**

x	a numeric vector giving the measurement value in the first dimension.
x_unit	the unit in which x was measured.
trans	a numeric vector giving the measurement value in the transition dimension.
trans_unit	the unit in which trans was measured.
y	a numeric vector giving the measurement value in the second dimension.
y_unit	the unit in which y was measured.

**Details**

This function supports all dimensions in `conv_unit_options` except for coordinates. The conversion values have been defined based primarily from international weight and measurement authorities (e.g. General Conference on Weights and Measures, International Committee for Weights and Measures, etc.). While much effort was made to make conversions as accurate as possible, you should check the accuracy of conversions to ensure that conversions are precise enough for your applications.

**Note**

**Duration** Years are defined as 365.25 days and months are defined as 1/12 a year.

**Energy** cal is a thermochemical calorie (4.184 J) and Cal is 1000 cal (kcal or 4184 J).

**Flow** All gallon-based units are US gallons.

**Mass** All non-metric units are based on the avoirdupois system.

**Power** hp is mechanical horsepower, or 745.69 W.

**Speed** mach is calculated at sea level at 15 °C.

**Author(s)**

Matthew A. Birk, <matthewabirk@gmail.com>

**See Also**

[conv\\_unit\\_options](#), [conv\\_unit](#)

**Examples**

```
# How many minutes does it take to travel 100 meters at 3 feet per second?
conv_dim(x = 100, x_unit = "m", trans = 3, trans_unit = "ft_per_sec", y_unit = "min")

# How many degrees does the temperature increase with an increase in 4 kPa given 0.8 Celcius
# increase per psi?
conv_dim(x_unit = "C", trans = 0.8, trans_unit = "C_per_psi", y = 4, y_unit = "kPa")

# Find the densities given volume and mass measurements.
conv_dim(x = c(60, 80), x_unit = "ft3", trans_unit = "kg_per_l", y = c(6e6, 4e6), y_unit = "g")
```

---

 conv\_unit

---

*Convert Units of Measurement*


---

**Description**

DEPRECATED. Converts common units of measurement for a variety of dimensions. See [conv\\_unit\\_options](#) for all options.

**Usage**

```
conv_unit(x, from, to)
```

**Arguments**

x	a numeric vector giving the measurement value in its original units.
from	the unit in which the measurement was made.
to	the unit to which the measurement is to be converted.

**Details**

**Acceleration** mm\_per\_sec2, cm\_per\_sec2, m\_per\_sec2, km\_per\_sec2, grav, inch\_per\_sec2, ft\_per\_sec2, mi\_per\_sec2, kph\_per\_sec, mph\_per\_sec

**Angle** degree, radian, grad, arcmin, arcsec, turn

**Area** nm2, um2, mm2, cm2, m2, hectare, km2, inch2, ft2, yd2, acre, mi2, naut\_mi2

**Coordinate** dec\_deg, deg\_dec\_min, deg\_min\_sec (see note)

**Count** nmol, umol, mmol, mol

**Duration** nsec, usec, msec, sec, min, hr, day, wk, mon, yr, dec, cen, mil, Ma

**Energy** J, kJ, erg, cal, Cal, Wsec, kWh, MWh, BTU

**Flow** ml\_per\_sec, ml\_per\_min, ml\_per\_hr, l\_per\_sec, l\_per\_min, l\_per\_hr, m3\_per\_sec, m3\_per\_min, m3\_per\_hr, gal\_per\_sec, gal\_per\_min, gal\_per\_hr, ft3\_per\_sec, ft3\_per\_min, ft3\_per\_hr, Sv

**Length** angstrom, nm, um, mm, cm, dm, m, km, inch, ft, yd, fathom, mi, naut\_mi, au, light\_yr, parsec, point

**Mass** ug, mg, g, kg, Pg, carat, metric\_ton, oz, lbs, short\_ton, long\_ton, stone

**Power** uW, mW, W, kW, MW, GW, erg\_per\_sec, cal\_per\_sec, cal\_per\_hr, Cal\_per\_sec, Cal\_per\_hr, BTU\_per\_sec, BTU\_per\_hr, hp

**Pressure** uatm, atm, Pa, hPa, kPa, torr, mmHg, inHg, mbar, bar, dbar, psi

**Speed** mm\_per\_sec, cm\_per\_sec, m\_per\_sec, km\_per\_sec, inch\_per\_sec, ft\_per\_sec, kph, mph, km\_per\_day, mi\_per\_day, knot, mach, light

**Temperature** C, F, K, R

**Volume** ul, ml, dl, l, cm3, dm3, m3, km3, us\_tsp, us\_tbsp, us\_oz, us\_cup, us\_pint, us\_quart, us\_gal, inch3, ft3, mi3, imp\_tsp, imp\_tbsp, imp\_oz, imp\_cup, imp\_pint, imp\_quart, imp\_gal

The conversion values have been defined based primarily from international weight and measurement authorities (e.g. General Conference on Weights and Measures, International Committee for Weights and Measures, etc.). While much effort was made to make conversions as accurate as possible, you should check the accuracy of conversions to ensure that conversions are precise enough for your applications.

**Note**

**Duration** Years are defined as 365.25 days and months are defined as 1/12 a year.

**Coordinate** Values must be entered as a string with one space between subunits (e.g. 70° 33' 11" = "70 33 11").

**Energy** cal is a thermochemical calorie (4.184 J) and Cal is 1000 cal (kcal or 4184 J).

**Flow** All gallon-based units are US gallons.

**Mass** All non-metric units are based on the avoirdupois system.

**Power** hp is mechanical horsepower, or 745.69 W.

**Speed** mach is calculated at sea level at 15 °C.

**Author(s)**

Matthew A. Birk, <matthewabirk@gmail.com>

**See Also**

[conv\\_unit\\_options](#), [conv\\_dim](#)

**Examples**

```
conv_unit(2.54, "cm", "inch") # Result = 1 inch

conv_unit(seq(1, 10), "kg", "short_ton") # A vector of measurement values can be converted

# Convert 1, 10, and 100 meters to all other length units
sapply(conv_unit_options$length, function(x) conv_unit(c(1, 10, 100), "m", x))

conv_unit("33 1 1", "deg_min_sec", "dec_deg")

conv_unit(c("101 44.32", "3 19.453"), "deg_dec_min", "deg_min_sec")
```

---

conv_unit_options	<i>Unit of Measurement Conversion Options</i>
-------------------	---

---

**Description**

DEPRECATED. Shows what units of measurement can be converted with the function [conv\\_unit](#).

**Usage**

```
conv_unit_options
```

**Format**

A list with all units available for conversion using [conv\\_unit](#).

**Details**

**Duration** Years are defined as 365.25 days and months are defined as 1/12 a year.

**Coordinate** Values must be entered as a string with one space between subunits (e.g. 70° 33' 11" = "70 33 11").

**Energy** cal is a thermochemical calorie (4.184 J) and Cal is 1000 cal (kcal or 4184 J).

**Mass** All non-metric units are based on the avoirdupois system.

**Power** hp is mechanical horsepower, or 745.69 W.

**Speed** mach is calculated at sea level at 15 °C.

**Author(s)**

Matthew A. Birk, <matthewabirk@gmail.com>

**Source**

The conversion values have been defined based primarily from international weight and measurement authorities (e.g. General Conference on Weights and Measures, International Committee for Weights and Measures, etc.). While much effort was made to make conversions as accurate as possible, you should check the accuracy of conversions to ensure that conversions are precise enough for your applications.

**See Also**

[conv\\_unit](#)

**Examples**

```
conv_unit_options
conv_unit_options$pressure
```

---

geom\_mean

*Geometric Mean*

---

**Description**

Computes the geometric mean of a vector,  $x$ . It is a wrapper for  $\exp(\text{mean}(\log(x)))$ .

**Usage**

```
geom_mean(x, add0.001 = FALSE, ignore_neg = FALSE, ...)
```

**Arguments**

<code>x</code>	a numeric vector or an R object which is coercible to one by <code>as.vector(x, "numeric")</code> .
<code>add0.001</code>	logical. Should a small constant (0.001) be added to avoid issues with zeroes?
<code>ignore_neg</code>	logical. Should negative values be ignored to avoid NaNs?
<code>...</code>	further arguments passed to <a href="#">mean</a> .

**Author(s)**

Matthew A. Birk, <matthewabirk@gmail.com>

**See Also**

[mean](#)

**Examples**

```
geom_mean(1:10)
geom_mean(0:10)
geom_mean(0:10, add0.001 = TRUE)
geom_mean(-10:10, add0.001 = TRUE, ignore_neg = TRUE)
```

---

range\_seq

*Sequence Generation Spanning A Numerical Range*

---

**Description**

Generates a sequence of numbers spanning the range of x.

**Usage**

```
range_seq(x, extend = 0, ...)
```

**Arguments**

x	a numeric vector.
extend	number specifying the fraction by which the range should be extended.
...	further arguments to be passed to <a href="#">seq</a> .

**Author(s)**

Matthew A. Birk, <matthewabirk@gmail.com>

**See Also**

[seq](#), [extendrange](#)

**Examples**

```
range_seq(rnorm(10, sd = 20))
range_seq(c(3, 9), extend = 0.1)
range_seq(c(3, 9), length.out = 20)
```

---

se	<i>Standard Error</i>
----	-----------------------

---

**Description**

Computes the standard error of the values in `x`. If `na.rm` is TRUE then missing values are removed before computation proceeds.

**Usage**

```
se(x, na.rm = FALSE)
```

**Arguments**

<code>x</code>	a numeric vector or an R object which is coercible to one by <code>as.vector(x, "numeric")</code> .
<code>na.rm</code>	logical. Should missing values be removed?

**Author(s)**

Matthew A. Birk, <matthewabirk@gmail.com>

**See Also**

[sd](#), [var](#)

**Examples**

```
se(1:10)
```

---

summ_stat	<i>Pooled Summary Descriptive Statistics</i>
-----------	--

---

**Description**

Pools summary statistics when given mean and (optionally) a measurement of variability (choose one among `var`, `sd`, and `se`).

**Usage**

```
summ_stat(mean, n, var, sd, se)
```

**Arguments**

mean	numeric. A vector of mean values to be pooled.
n	numeric. A vector of n values to be pooled.
var	numeric. A vector of variance values to be pooled.
sd	numeric. A vector of standard deviation values to be pooled.
se	numeric. A vector of standard error of the mean vlaues to be pooled.

**Author(s)**

Matthew A. Birk, <matthewabirk@gmail.com>

**See Also**

[weighted.mean](#), [se](#)

**Examples**

```
summ_stat(mean = c(0.68, 0.67), n = c(4, 5), sd = c(0.11, 0.15))
summ_stat(mean = 0.68, n = 3, se = 5)
summ_stat(mean = rnorm(1e4), n = rep(1, 1e4)) # Find pooled mean when variability is unknown.
```

---

which.closest	<i>Where is the closest?</i>
---------------	------------------------------

---

**Description**

Returns index of the closest value to x.

**Usage**

```
which.closest(vec, x)
```

**Arguments**

vec	a numeric vector.
x	numeric. The value for which the closest match should be returned.

**Author(s)**

Matthew A. Birk, <matthewabirk@gmail.com>

**See Also**

[which.min](#), [which.max](#)

10

*which.closest*

### **Examples**

```
which.closest(10:1, 3.3)
```

# Index

## \* datasets

conv\_unit\_options, 5

birk, 2

birk-package (birk), 2

conv\_dim, 2, 5

conv\_unit, 3, 3, 5, 6

conv\_unit\_options, 2, 3, 5, 5

extendrange, 7

geom\_mean, 6

mean, 6

range\_seq, 7

sd, 8

se, 8, 9

seq, 7

summ\_stat, 8

var, 8

weighted.mean, 9

which.closest, 9

which.max, 9

which.min, 9